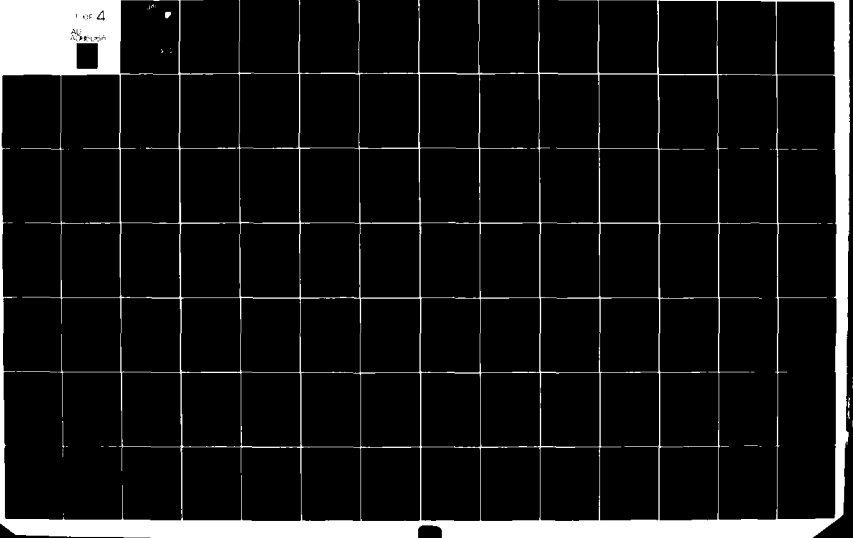
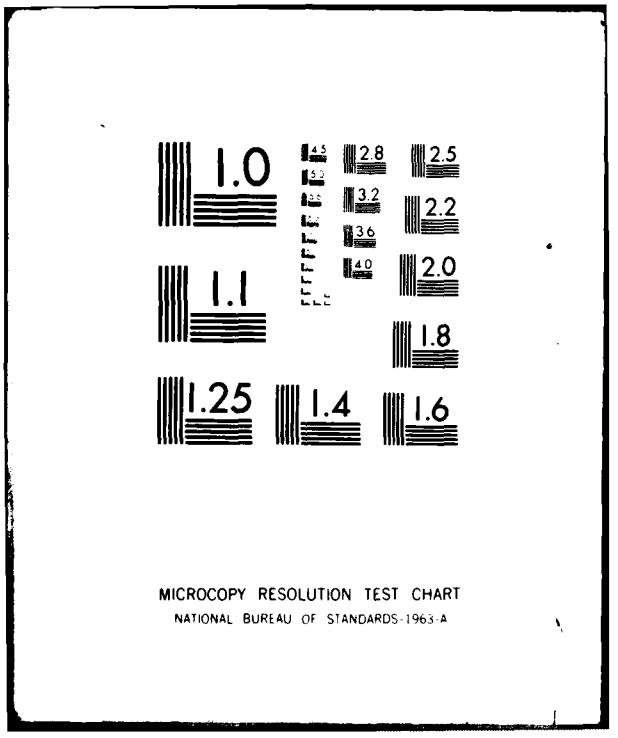
AD-A086 066 HARRIS CORP MELBOURNE FL GOVERNMENT COMMUNICATION SY--ETC F/G 9/2
MULTIPLE MICROPROCESSOR SYSTEM (MMS) DESIGN STUDY. VOLUME II.(U)
MAR 80 F30602-78-C-0114
UNCLASSIFIED RADC-TR-80-33-VOL-2 NL

1 OF 4
AD
A086066




1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

LEVEL III

(12)

SC

AD A086066

RADC-TR-80-33, Vol II (of four)
(9) Final Technical Report,
(11) March 1980

ADA086066

(6) MULTIPLE MICROPROCESSOR SYSTEM (MMS) DESIGN STUDY. Volume II.

Harris Corporation

Government Communications Systems Division

(18) RADC (19) TR-80-33-VOL-2

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

(12) 362

DTIC
ELECTE
JUL 1 1980
S D
A

(15) F30602-78-C-0114

ROME AIR DEVELOPMENT CENTER
Air Force Systems Command
Griffiss Air Force Base, New York 13441

411224 Am

80 6 30 017

This report has been reviewed by the RADC Public Affairs Office (PA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

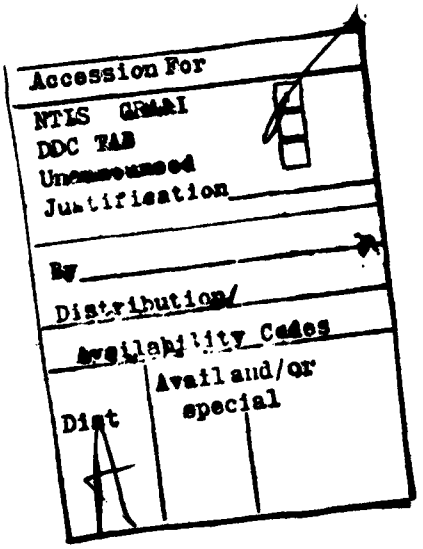
RADC-TR-80-33, Volume II (of four) has been reviewed and is approved for publication.

APPROVED: Michael A. Troutman

MICHAEL A. TROUTMAN, 1Lt, USAF
Project Engineer

APPROVED: Wendall C. Bauman

WENDALL C. BAUMAN, Colonel, USAF
Chief, Information Sciences Division



FOR THE COMMANDER: John P. Huss

JOHN P. HUSS
Acting Chief, Plans Office

If your address has changed or if you wish to be removed from the RADC mailing list, or if the addressee is no longer employed by your organization, please notify RADC (ISCA) Griffiss AFB NY 13441. This will assist us in maintaining a current mailing list.

Do not return this copy. Retain or destroy.

F30602-78-C-0114

TABLE OF CONTENTS

Section	Title	Page
10.0	MMS FUNCTIONAL DESCRIPTION ----------	10-1
10.1	Glossary ----------	10-7
10.2	Functional Description-Master Level---------	10-9
10.2.1	Segment 5 Bus Controller (S5BC) ----------	10-9
10.2.2	Facility Control Processor Interface (FCPI)-	10-10
10.2.3	Input-Output Interface/Input-Output Processor (IOP)----------	10-12
10.2.4	Master Broadcast Controller (MBC) ----------	10-15
10.2.5	Performance Monitoring System Interface/ Performance Monitoring Interface/PMS Interface (PMSI) ----------	10-16
10.2.6	Shared Resource Controller (SRC) ----------	10-21
10.2.7	Time Alignment Controller (TAC) ----------	10-25
10.3	Functional Description - Segment Level -----	10-26
10.3.1	Segment n Control and BIU (SnCBIU) ---------	10-26
10.3.2	Broadcast Controller (BC) ----------	10-27
10.4	Functional Description - PE Level ----------	10-28
10.4.1	Bus Interface Unit (BIU) ----------	10-28
10.4.2	Arbitration ----------	10-29
10.4.3	Memory Interface Unit (MIU)/Memory Interface (MI) ----------	10-30
10.4.4	Local Memory (LM) ----------	10-32
10.4.5	Instruction Execution Unit (IEU/IEU&IOC)----	10-33
10.4.6	Control Store (CS) ----------	10-37
10.4.7	Internal Memory (IM) ----------	10-39
10.4.8	Pseudo-Time Accumulator ----------	10-40
10.5	Abbreviations Used in Section 10 ----------	10-41
	Last Page ----------	10-42
11.0	OVERVIEW - MMS SOFTWARE DESIGN ----------	11-1
11.1	Abbreviations Used In Section 11 ----------	11-8
	Last Page ----------	11-8
12.0	MMS DIAGNOSTIC PROGRAMS ----------	12-1
	AØ - HARDWARE DIAGNOSTICS (FROM FCP VIEW) --	12-8
	AØ - HARDWARE DIAGNOSTICS (FROM PE VIEW) ---	12-10
	AØ - HARDWARE DIAGNOSTICS (FROM "BIG 7" VIEW)	12-12

TABLE OF CONTENTS (Cont'd)

Section	Title	Page
12.1	Abbreviations Used In Section 12 ----------	12-14
	Last Page ----------	12-14
13.0	E-CODE COMPILER ----------	13-1
13.1	Abbreviations Used in Section 13 ----------	13-2
	Last Page ----------	13-2
14.0	MICROCODE ASSEMBLER ----------	14-1
	CØ - MICROCODE ASSEMBLER ----------	14-4
	C1A - SET UP SYMBOL TABLE ----------	14-8
	C2A1 - BUILD IDENTIFICATION SECTION ---------	14-10
	C2A2 - PROCESS MICROCODE INSTRUCTIONS -------	14-13
	C1B - PROCESS LABEL ----------	14-15
	C1C - PROCESS OPCODE ----------	14-17
	C1D - PROCESS OPERAND ----------	14-19
14.1	Abbreviations Used in Section 14 ----------	14-21
	Last Page ----------	14-21
15.0	PE EXECUTIVE (PEXE) ----------	15-1
	DØ - PEXE ----------	15-3
	D1A - INITIALIZATION OF DATA ----------	15-5
	D1B - DEBUG MICROCODE ----------	15-8
	D1C - PROCESS DEBUG/PMS INTERRUPT ----------	15-10
	D1D - PROCESS ERROR INTERRUPT ----------	15-12
	D1E - PROCESS H/W INTERVAL TIMER INTERRUPT --	15-14
	D2E5 - EXECUTE SOFTWARE ROUTING ----------	15-17
	D1F - PROCESS I/O REQUEST ----------	15-19
	D1G - EXECUTE SCHEDULER ----------	15-21
	D1H - PROCESS PSEUDO-CLOCK ALIGNMENT --------	15-23
	D1I - DO CLEAN UP ----------	15-25
15.1	Abbreviations Used in Section 15 ----------	15-27
	Last Page ----------	15-27
16.0	SYSTEM GENERATION LOADER ----------	16-1
	EØ - SYSTEM GENERATION LOADER (SGL)----------	16-4

TABLE OF CONTENTS (Cont'd)

Section	Title	Page
	E1A - EXECUTE STARTUP	16-6
	E1B - CONFIGURE TS	16-8
	E2B1 - SELECT TS FROM FILE	16-11
	E2B2 - SELECT HARDWARE MODULES	16-13
	E2B3 - LINK IN PEXE	16-15
	E2B4 - LINK IN INTERCONNECTS	16-17
	E1C - SELECT TS OPERATING SYSTEM	16-20
	E1D - CONFIGURE "BIG 7"	16-22
16.1	Abbreviations Used in Section 16	16-24
	Last Page	16-24
17.0	RUN-TIME EXECUTIVE	17-1
	FØ - RUN-TIME EXECUTIVE (RTE)	17-6
	F1A - CONFIGURE ENVIRONMENT	17-11
	F2A1 - SET UP I/O DESCRIPTOR	17-13
	F2A2 - ESTABLISH HANDLER FOR I/O	17-15
	F2A3 - ALLOCATE DUAL BUFFER FOR I/O	17-17
	F2A4 - FILL DUAL BUFFER	17-19
	F2A5 - STORE CONNECTING ADDRESS IN IOP	17-21
	F1B - CONFIGURE DEBUG	17-23
	F2B1 - CHOOSE PROCESSOR (PE) FOR DEBUG	17-25
	F2B2 - CHOOSE LEVEL	17-27
	F2B3 - SELECT OPTIONS	17-29
	F2B4 - SELECT OPTIONS	17-32
	F2B5 - UPDATE DEBUG FLAGS	17-33
	F2B6 - UPDATE LINKAGE TABLES	17-35
	F2B7 - MODIFY PE MICROCODE	17-37
	F1C - CONFIGURE PMS	17-39
	F2C1 - CONFIGURE PMS SAMPLING	17-41
	F2C2 - CONFIGURE PMS STORAGE FILE	17-42
	F2C3 - CONFIGURE RUN-TIME DISPLAY	17-44
	F1D - EXECUTE TS	17-46
	F2D1 - INITIALIZE TARGET SYSTEM	17-48
	F2D2 - PROCESS DEBUG	17-50

TABLE OF CONTENTS (Cont'd)

Section	Title	Page
	F2D3 - PROCESS PMS	17-52
	F2D4 - EXECUTE DATA OUTPUT HANDLER	17-54
	F2D5 - EXECUTE DATA INPUT HANDLER	17-56
	F2D6 - TIDY UP	17-58
17.1	Abbreviations Used In Section 17	17-60
	Last Page	17-60
18.0	POST MORTEM DUMP	18-1
	GØ - POST MORTEM DUMP	18-4
	G1B - LOCATE ADDRESS	18-6
18.1	Abbreviations Used In Section 18	18-9
	Last Page	18-9
19.0	PMS OFF-LINE ANALYSIS	19-1
19.1	Display of Data	19-3
19.1.1	Table	19-4
19.1.2	Scattergram or Scatter Diagram	19-4
19.1.3	Histogram	19-5
19.1.4	Table Printouts from Statistical Analysis Routines	19-5
19.1.5	Line Fits from Fitting Data to Equations	19-6
19.2	Statistical Analysis	19-7
19.2.1	Frequency Analysis	19-8
19.2.2	Central Tendency	19-8
19.2.3	Measure of Variation	19-9
19.3	Line Fits	19-11
19.3.1	Broken Line	19-12
19.3.2	Straight Line by Selected Points	19-12
19.3.3	Straight Line by Least Squares	19-13
19.3.4	Second Degree Parabola by Least Squares	19-13
19.4	General Routines	19-15
19.4.1	Summary	19-16
19.4.2	Frequency	19-16
19.5	References	19-17
19.6	Top-Down Design	19-18
	HØ - PMS OFF-LINE ANALYSIS	19-20
	H1A - SELECT ROUTINE	19-22
	H1B - EXECUTE ROUTINE	19-24
	H1C - DISPLAY RESULTS	19-26

TABLE OF CONTENTS (Cont'd)

Section	Title	Page
19.7	Abbreviations Used In Section 19 ----------	19-28
	Last Page ----------	19-28
20.0	TARGET SYSTEM ----------	20-1
20.1	Emulation Modules ----------	20-2
	IØ - TARGET SYSTEM EMULATION ----------	20-4
	I1A - IDENTIFY ID OF H/W EMULATOR ----------	20-6
	I1B - IDENTIFY TS USER INPUTS ----------	20-8
	I1C - IDENTIFY STATUS WORDS ----------	20-10
	I1D - IDENTIFY I/O BUFFERS ----------	20-12
	I1E - IDENTIFY I/O PORTS ----------	20-14
	I1F - IDENTIFY TIMING ----------	20-16
	I1G - CREATE IEC MICROCODE MODULE ----------	20-18
	I1H - CREATE H/W MICROCODE MODULE ----------	20-20
20.2	Proposed Target System ----------	20-22
20.3	Emulation of PDP-11/45 ----------	20-25
20.3.1	Emulation Modifications ----------	20-25
20.3.2	Instruction Emulation Code (IEC) ----------	20-26
20.3.3	Disk Configuration ----------	20-29
	Disk Emulation Routine ----------	20-30
20.3.4	Magnetic Tape Configuration ----------	20-31
	Mag Tape Emulation Routine ----------	20-32
20.3.5	PDP to Intel Communication Configuration - "BUS G" ----------	20-33
20.4	Emulation of Intel 8080 ----------	20-34
20.4.1	Instruction Emulation Code (IEC) ----------	20-34
20.4.2	Shared Memory ----------	20-38
20.4.3	Intel 8080 to PDP-11/45 Communication - "BUS G" ----------	20-40
20.5	Register Emulation ----------	20-41
20.6	Abbreviations Used In Section 20 ----------	20-43
	Last Page ----------	20-43
21.0	USER SCENARIO ----------	21-1
21.1	"Help" File ----------	21-8
21.2	SGL (System Generation Loader) ----------	21-11
21.2.1	Interactive Mode ----------	21-12
21.2.2	Quick Mode ----------	21-38
21.2.3	Batch Mode ----------	21-41
21.2.4	Loading of Existing TS ----------	21-43
21.3	RTE (Run-Time Executive) ----------	21-46
21.4	PMD (post Mortem Dump) ----------	21-55
21.5	PMS (PMS Off-Line Analysis)----------	21-56
21.6	Abbreviations Used In Section 21 ----------	21-57
	Last Page ----------	21-57

TABLE OF CONTENTS (Cont'd)

Section	Title	Page
22.0	FCP DESCRIPTION ----------	22-1
	Last Page	22-4
23.0	GENERAL TOPICS ----------	23-1
23.1	Synchronous Busing Structure ----------	23-2
23.2	Executive for IOP ----------	23-4
23.3	Executive for PMSI ----------	23-6
23.4	Executive for SRC ----------	23-8
23.5	Abbreviations Used in Section 23	23-11
	Last Page	23-11

LIST OF ILLUSTRATIONS

Figure	Title	Page
10.0-1	MMS Functional Structure	10-2
10.0-2	Segment Configuration (Segments 1-4)	10-4
10.0-3	PE Configuration	10-5
10.2.6-1	Suggested Contents for Message Handling	10-22
10.2.6-2	Suggested Contents for Resource Control Block (RCB)	10-23
10.2.6-3	Suggested Contents for Process Control Block (PCB)	10-23
10.4.3	Memory Map Table Format	10-30
10.4.6	Proposed Format for Microcode Word	10-38
11.0-1	Structured Analysis Module (SA)	11-2
11.0-2	Top Level Design of MMS	11-3
11.0-3	Detailed Modular Design of MMS	11-5
12.0-1	Top-Down Design Chart	12-2
12.0-2	Diagnostic Package Interactions	12-3
14.0-1	Top-Down Design Charts	14-3
14.0-2	Proposed Format of Object Module created by Microassembler	14-5
15.0-1	Top-Down Design Chart	15-2
15.0-2	Suggested Format for Emulation Modules	15-6
16.1	Top-Down Design Chart	16-3
17.0-1 a	Top-Down Design Chart	17-3
17.0-1 b	Top-Down Design Chart	17-3
17.0-1 c	Top-Down Design Chart	17-4
17.0-1 d	Top-Down Design Chart	17-4
17.0-1 e	Top-Down Design Chart	17-5
17.0-2	RTE I/O Descripter Record Format	17-8
17.0-3	RTE DEBUG/PMS Option Table Format	17-9
18.0	Top-Down Design Chart	18-3
19.6	Top-Down Design Chart	19-19
20.1	Top-Down Design Chart	20-3
20.2	TS To Be Emulated	20-23
20.3.2-1	Beginning Details of the IEC for PDP-11/45	20-27
20.3.2-2	Emulation of Some PDP-11/45 Commands	20-28
20.4.1-1	Beginning Details of the IEC for INTEL 8080	20-35
20.4.1-2	INTEL 8080 FETCH Routine	20-56
20.4.1-3	Details of Some INTEL Commands	20-57

LIST OF ILLUSTRATIONS (Cont'd)

Figure	Title	Page
21.0-1	Detailed Modular Design of MMS	21-2
21.0-2	Top-level Design of SGL	21-4
21.0-3	Top-level Design of RTE	21-5
21.0-4	Top-level Design of PMD	21-6
21.0-5	Top-level Design of PMS	21-7
21.1	"Help" File	21-10
21.2.1-1	Interactive Mode Scenario	21-13
to 21.2.1-25		to 21-37
21.2.2-1	Quick Mode Scenario	21-39
21.2.2-2	Quick Mode Scenario	21-40
21.2.3	Batch Mode Scenario	21-42
21.2.4-1	TS Quick Mode Scenario	21-44
21.2.4-2	TS Quick Mode Scenario	21-45
21.3-1	RTE Interactive Displays	21-47
to 21.3-8		to 21-54

10.0 MMS FUNCTIONAL DESCRIPTION

This section reviews all of the functional components of MMS, and defines those areas where software is needed or where software interfaces.

MMS consists of a network of up to 64 (sixty-four) microprocessors linked to a minicomputer. Figure 10.0-1 is the functional block diagram used to illustrate this network. The FCP (Facility Control Processor) within Figure 10.0-1 is the mini-computer. Several computers are evaluated relative to the needs of MMS, and this study is presented in Chapter 22, "FCP DESCRIPTION" .

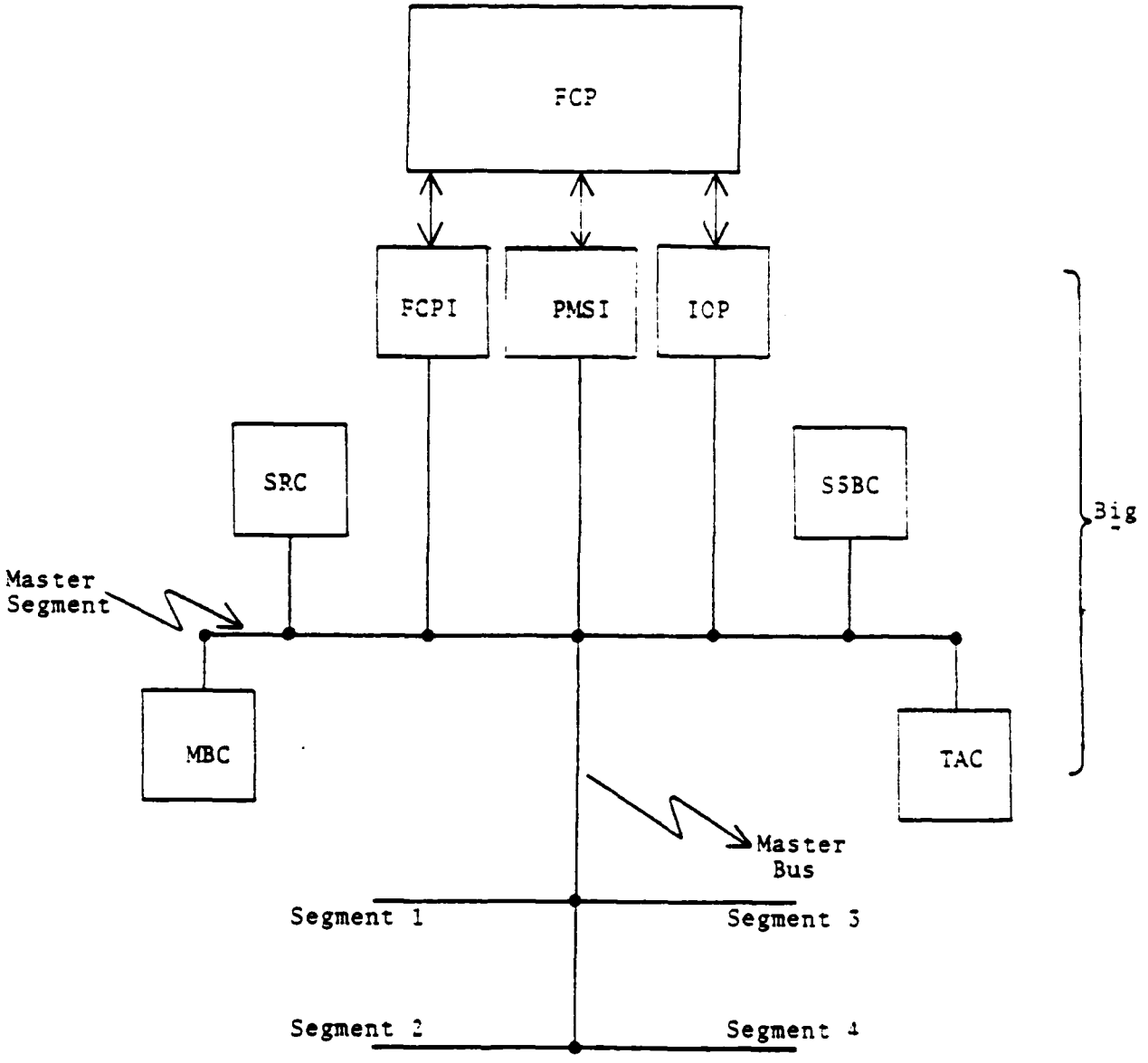
Functionally, the FCP stores, on its disks, the libraries of emulated modules, emulated computer systems, and emulated target systems. The FCP allows the user to configure his target system onto the MMS hardware, and to interface with his emulated target system during its execution.

Once the functional description is covered within this section, the software necessary for MMS is covered in the next section, "Overview - MMS Software" (Section 11).

Again, in reference to Figure 10.0-1, three components link in directly to the FCP. These are:

- FCPI - Facility Control Processor Interface
- PMSI - Performance Monitoring System Interface
- IOP - Input/Output Processor

The FCPI is used for downloading large blocks of information to a PE (Processing Element). Error messages will also be processed in both directions through FCPI. Thus, a PE can send error messages to the FCP through FCPI.



MMS FUNCTIONAL STRUCTURE

Figure 10.0-1

PMSI is used to pass performance monitoring data from any PE to the user at the FCP. PMSI has to process run-time data differently than it processes off-line data. Off-line data is saved on a disk for later processing. Run-time data is processed immediately and some type of results are printed on the user's CRT (Cathode Ray Tube, which in an interactive terminal).

The IOP allows the TS (Target System) to link to several types of data generators residing within, or through, the FCP. These dated generators can be a file of data values previously generated, or can be a program to generate new values whenever the program is accessed, or can be some real device which is linked through the FCP.

Four other components reside on this Master Segment. These are: MBC (Master Broadcast Controller), SRC (Shared Resource Controller), S5BC (Segment 5 Bus Controller), and TAC (Time Alignment Controller). These seven components comprise the master components utilized by all PE's. Thus, they will be referred to as the "Big 7", to indicate that a process is to be done here to aid the PE's.

The Master Bus has four additional segments. Each segment has a SnCBIU (Segment n Control and Bus Interface Unit), a BC (Broadcast Controller), and sixteen PE's, as illustrated by Figure 10.0-2. Figure 10.0-3 gives a further functional description of these PE's.

All of these units - for the Master Bus, for a segment, and for a PE - are separately discussed in the following pages. Section 10.2 will cover the descriptions for the "Big 7". Section 10.3 will cover the descriptions for the segment, and Section 10.4 will be that for the PE's.

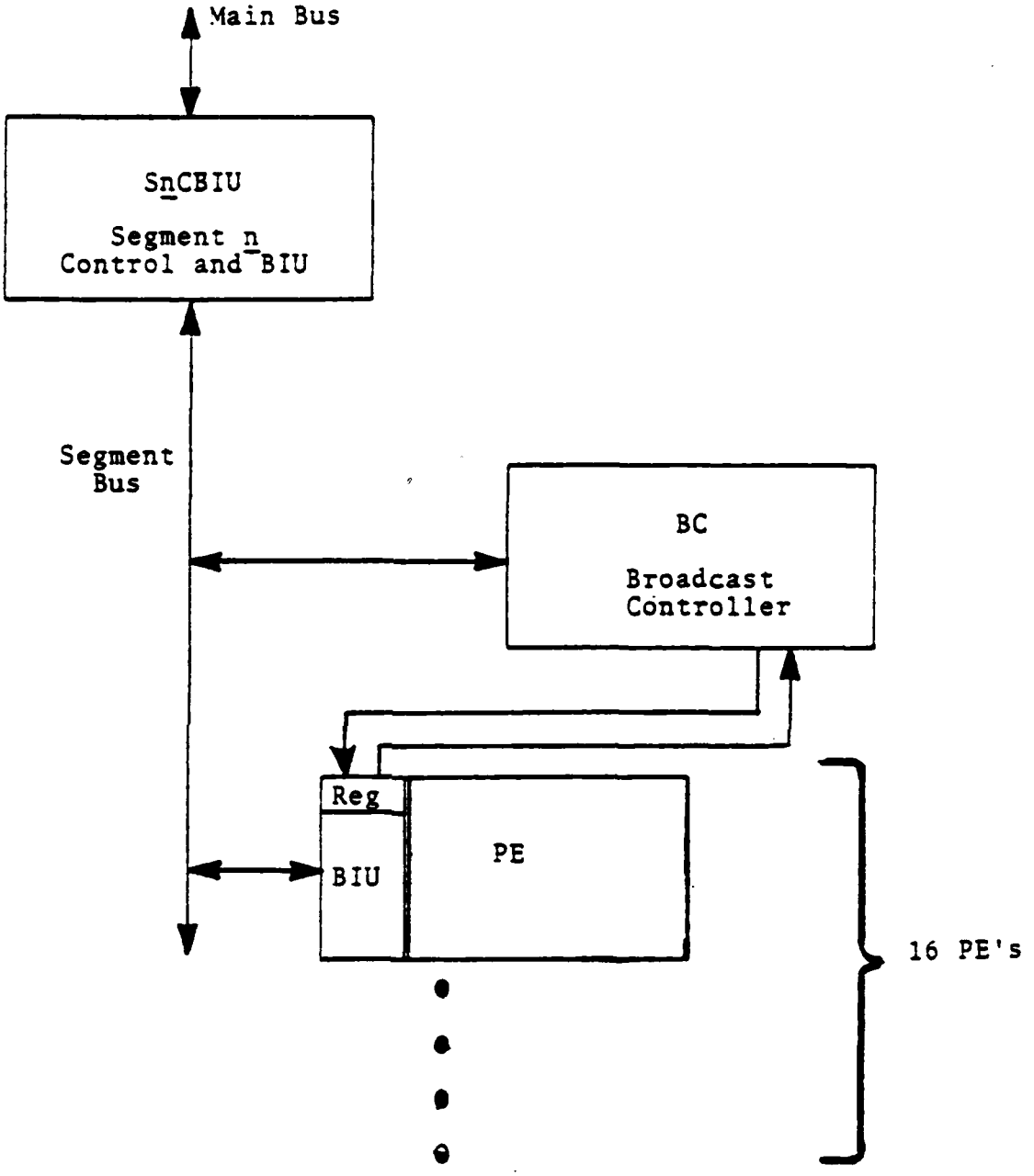


Figure 10.0-2 Segment Configuration
(Segments 1-4)

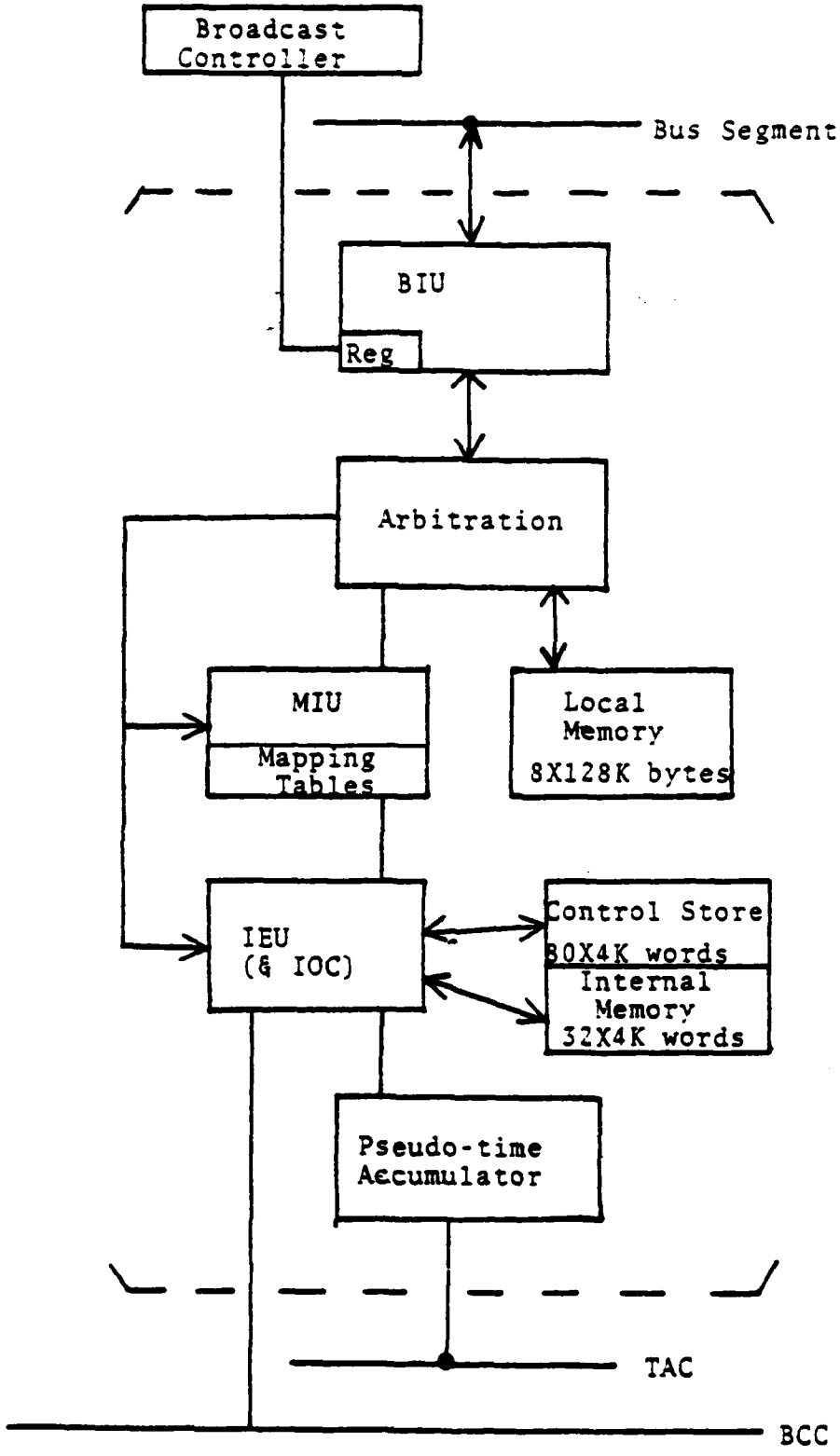


Figure 10.0-3 PE Configuration

To further aid the user, a glossary of terms is included as Section 10.1. The intent of this glossary is to define the scope of some of the terminology of the software document. Section 10.5 sets a standard for all following major sections in that it contains an alphabetical definition of all abbreviations used within Section 10.

10.1 Glossary

TIME

Real Time is the same as Wall Clock. This refers to the execution of all parts of the target system as "real world". Pseudo-time is the MMS approximation to Wall Clock time. Since all parts are executing synchronously, this time can be slowed down to study the target system while still maintaining "real time" conditions, i. e., time coherence.

PROGRAMMER

It is visualized that two levels of programmers are necessary for MMS utilization. The persons that develop the emulator packages will need detailed knowledge of the hardware of the peripherals and of the computer. This person could use the Microcode Assembler, and thus be referenced as the Microcode Programmer. If the E-Code compiler is available, these emulation packages could be developed on this system. Thus, this person could be known as the Emulator Programmer. The general User or general Programmer uses the library of emulators developed by the Emulator Programmer to configure a target system.

DATA COLLECTION

Two types of data are referenced. During the execution of MMS, the data collection is the same. If the user wishes to display the results while the system is executing, this is called run data, real-time data, or active data. If the user wishes to store the results on a disk file for later analysis, this is called off-line data or static data.

VENDOR

This term is used several places to refer to software supplied by the company which writes the MMS operating software. If the term user is used, then this refers to software that the vendor does not supply.

10.2 Functional Description-Master Level

The Master Segment contains the "Big 7" modules. Each is separately described. (See Figure 10.0.2).

10.2.1 Segment 5 Bus Controller (S5BC)

Purpose: S5BC will arbitrate bus requests and will decide which segment ("Segment n Control and BIU" unit) will gain control over the bus.

S/W: None required.

10.2.2 Facility Control Processor Interface (FCPI)

Purpose: The FCPI is used to download large amounts of information needed to configure the TS and is used for the transfer of one word messages in either direction between the FCP and the TS.

Theory: When the user is configuring the parts of the TS, the following modules/data are downloaded to the PE's:

- PEXE (PE Executive)
- IEC (Instruction Emulation Code)
- Emulated Hardware modules
- TS Software (Operating System and User Programs)
- Registers
- Control information

When the user is starting execution of the TS, or when the user wishes to interact with the TS during execution or with DEBUG/PMS, these one-word messages are processed through FCPI. Whenever an error condition develops through the execution of the TS, these error messages will be passed to FCP through FCPI.

S/W: No software is required. All transfers between FCP and PEXE (or other components of MMS) are to be DMA, with the ability to also set up an interrupt within FCP for the immediate processing of error conditions.

Note: The following information is needed for FCPI to process data or files:

- Type of process
  - 01 = Input to PE
    10 = Output from PE (DMA)
    11 = Output from PE (Interrupt)
    00 = Verify block of information

- Address of buffer in FCP
- Address of buffer in memory of PE
- Word Count (This count should be a multiple of 8 so that MMS can emulate computer systems which have 8 bits/word.)

10.2.3 Input-Output Interface/Input-Output Processor (IOP)

Purpose: The IOP resolves linkage between the TS emulated within MMS and the physical files and/or programs residing within the FCP to satisfy environmental data.

Theory: When a TS is configured, the linkage to an external program, file, or device is initiated by the System Generation Loader (SGL) phase. This linkage is from TS to IOPT (Input/Output Port Table) within IOP. When environmental linkages are defined within the RTE (Run-time Executive) phase, IOPT will have linkages defined to the FCP program, file, or device. An executive is to be vendor-supplied, which will have the function of resolving priorities of those accesses to the FCP. Control over the pseudo-clock will not be a function of the IOP because the emulated hardware modules will retain this control.

Note: A PE will average an I/O access every 100 TS instructions. To keep the IOP from being a "weak link", all emulated processes should be partially local (handled by another PE) and partially centralized (handled within IOP). Thus, small to medium I/O could be handled by another PE, by storing the data file within the Internal Memory of a neighboring PE. Large amounts of data, or a program in the FCP which generates data, or an external device to the FCP would be processed through IOP.

H/W: Internal Memory of 64-bits X 4K. The IOPT (as part of software) is stored in this memory.

S/W:

1) IOPT, which is a table of at least 2,000 simplex ports. This is to be in an Internal Memory of 64-bits by 4K. The contents of this table should include:

- On/Off bit (or Busy/Not-busy flag)
- Priority
- Port number (relative to computer of TS)
- PE number
- I/O flag (Ø for Output from PE; 1 for Input to PE)
- Two words:
  - address(es) of buffer(s) within Internal Memory of PE
  - addresses of dual buffers within memory of FCP
  - address of program to be exercised plus one data word to be "seed" data into program (if applicable)
  - address of external device to FCP, which is to be accessed for data. Again, second word could be "seed" data.
  - current address of active input buffer
  - buffer length
  - counter
  - return address when data transfer is done

2) Microcode program to be "burned" into a PROM. This program will be accessed whenever an I/O process is to be handled. The emulated modules for I/O will access PEXE and will have selected the desired arbitration scheme within PEXE. The I/O arbitration scheme will activate the "ON" bit. Then the functions of this microcode program are:

a) Select I/O process of highest priority.

b) Do transfer one data word into IOP Data Register.

c) Do transfer of one data word from IOP Data Register into destination.

d) Set Interrupt within FCP if this is an Interrupt process rather than DMA process.

e) If buffer length has been reached, change current address to alternate.

f) Turn off "ON" bit.

g) Check to see of any other I/O processes are to be handled.

h) Loop back to step (a).

10.2.4 Master Broadcast Controller (MBC)

Purpose: The MBC accepts incoming messages from the FCP (Facility Control Processor) or from any PE (Processing Element), which is really from PEXE (PE Executive). The MBC then formulates that message with an address which reflects that the message is to be received by all PE's. Then the message is sent (broadcasted) to all PE's.

Theory: A one-word message is formulated in either FCP or in a PE (PEXE). The SBS (Synchronous Busing Structure, see Section 23.1) word format of the message contains the address of the register of MBC. When this register is then filled with this word (at least 48-bits wide), an interrupt signals the hardware within MBC. The "busy" flag is set to show that MBC is busy. The address portion of the register is changed to reflect that the outgoing message is for all PE's. Hardware will claim the bus, and then send the contents of the register. Hardware then releases the "busy" flag.

S/W: None required.

10.2.5 Performance Monitoring System Interface/Performance Monitoring Interface/PMS Interface (PMSI)

Purpose: The PMSI accepts DEBUG/PMS data from the PE's, processes the data to determine if for run-time analysis or if for off-line collection, and then routes the DEBUG/PMS data to designated area within FCP so that FCP can then process it.

Theory:

A. The classes of data are:

1. Run-time PMS data. When this data is stored within a buffer of FCP, which is large enough to contain one unit of PMS data, an interrupt is sent to the FCP so that the FCP can immediately process the routines to massage the data. If the PMS data is identifying a breakpoint, then PMSI will stop the pseudo-clock. Any responses to this breakpoint will be handled through FCPI to PEXE. The restart of the pseudo-clock will thus be handled by PEXE.

2. Off-line PMS data. This data is stored into one of the dual buffers within the FCP memory set up for the rapid collection of PMS data. This is a DMA process. The dumping of these buffers will be accomplished by the FCP (using one of its utilities). Since PMSI will have to monitor for "full" buffer in order to switch to the empty buffer, PMSI should alert the FCP, also, through an interrupt.

3. Both Run-time and off-line PMS. This means that a copy of the PMS data is sent through both processes (1) and (2).

4. Run-time DEBUG. The data is stored within a memory area of FCP, which area is large enough to hold one unit of DEBUG data. Then an interrupt is sent to the FCP so that it can

immediately process the information. If the DEBUG data identifies a breakpoint, PMSI will stop the pseudo-clock. Again, user responses will be handled through FCPI to the PE, and the restart signal to the pseudo-clock will initiate from PEXE.

5. Off-line DEBUG. Since off-line DEBUG data will look as if it were PMS data, the collection of DEBUG data will be the same as for PMS data. Thus, the intermixing of DEBUG and PMS (if the user so wishes), will be no problem.

6. Both Run -time and off-line DEBUG. This means that a copy of the DEBUG data is sent through both processes (4) and (5).

B. When the user specifies either DEBUG or PMS during RTE, a dual-buffer is set up in the memory of the FCP for data collection. An interrupt handler has to be available in the FCP to empty these buffers to disk whenever the FCP receives an interrupt from PMSI on the master segment of MMS; this is usually one of the utilities of the FCP. Linkage from the PMSI to these buffers has to be established. Also, the PMSI has to have a handler to remember in which of the two buffers it is currently storing data to be able to signal the FCP when one of them is full and to then be able to switch to the alternate buffer.

C. When PMSI receives the data from the PE, PMSI has to be able to determine the class of data received - whether to send to buffer, to send to a register with a signal to FCP to process register contents immediately (for run -time data), or to send data both ways.

D. The types of data which can be collected and how they are processed are:

1. Memory accesses - When user wishes to monitor one memory location or a block of memory, the memory span is located in the Memory Mapping Table of the MI of the PE; this is stored in the upper portion of the table. The memory map is then adjusted at the top section to delete reference to the section to be monitored. The reference for the monitored section is then stored separately in the Mapping Table with necessary monitor flags set.

2. Macro instruction - When the microcode program is written for the Instruction Emulation Code (IEC), a table will exist in Internal Memory which will contain addresses to Control Store (within the IEC routine) which are locations to emulate the different instructions. Since the word length of Internal Memory is 32-bits wide, and since only the left-most 16 bits are required for the address into Control Store, the right-most 16-bits are the PMS/DEBUG flags for the instructions. These flags are set by RTE under user control.

3. TS registers - Since these registers are part of Internal Memory, the right-most 16 bits are again used for PMS/DEBUG flags.

4. I/O accesses -

a. If the I/O process desired refers to all I/O processes, this would be handled on the instruction level.

b. If the I/O process desired refers to a specific device or handler, the flags should be included in the Address Table which refers to all emulated hardware modules. This is

also part of Internal Memory, and is used by programs in Control Store.

c. Shared Resource accesses will be monitored on the PE level. Memory will thus be monitored through the Memory Mapping Table in MIU. Other resources will be monitored through the emulated hardware module. This necessitates having SGL set flags within several PE's when referring to a shared resource.

5. Pseudo-time clock - When the user decides to set a breakpoint or to start a trace depending upon pseudo-time, an interrupt handler is required to process these conditions. Hardware Interval Timer 15 should be dedicated for this condition. Since the address table for these Interval Timers is located in Internal Memory, the right-hand 16-bits will contain the PMS/DEBUG flags. The Interval Timer is only 16-bits wide, so larger values will have to be handled within the software interrupt handler. Thus, intervals greater than 640 microseconds will have to be loops of resetting the Interval Timer until total desired time has elapsed.

6. Micro-instructions - No PMS monitoring is done of micro-instructions. Hardware is implemented within each PE to cause the single step execution of micro-instructions when in the DEBUG mode.

H/W: Need hardware to control storage of data into the PMSI register. Hardware talks to "calling" PE's. Hardware sends interrupt to PMSI executive when data is accepted within PMSI register.

S/W: A PMSI Executive (which will later become firmware) is needed to handle the DEBUG/PMS data that is generated through the execution of the TS. This Executive will stop TAC if the event is a breakpoint. If the event involves offline collection, then the Executive will send the data to a buffer in the FCP. If the event is to be run-time monitored, then the data is placed in a special buffer and an interrupt is sent to the FCP. Because of the interrupt, FCP will process and display the data.

10.2.6 Shared Resource Controller (SRC)

Purpose: The SRC handles the interfacing of a shared resource between two or more PE's of some emulated TS. The SRC will handle the arbitration between the users of each shared resource.

Theory: The SRC will be the same physical configuration as the other PE's of MMS. Several types of arbitration will be included in the Executive for this component. The user, when he designs the software to emulate some shared resource, will store part of the package within the PE and part of the software package in SRC. Then, when linkage is accomplished during the SGL phase, duplicate packages within SRC are eliminated. When the TS wishes to utilize a shared resource, it will link to its module within SRC. This microcode package will resolve arbitration of who gets to use the shared resource. The handling of data is accomplished through known linkage paths to emulated shared resource. Then control is returned to initiating modules within the PE. Timing is established and controlled by SRC in order to reflect arbitration.

S/W: Software needed (will become firmware):

a) Executive in microcode with table to link between modules within PE's and the linkage module within SRC.

b) Executive should have the function of granting control to those shared resource emulations of highest priority.

c) Arbitration subroutines in microcode which are available for user to utilize.

d) Executive must have ability to accept messages from

PE's, sending "grant" messages to PE's, sending results to PE's, and then accepting "release" messages from the PE's.

e) Executive should have ability to send the pseudo-time elapsed, if arbitration is based upon total delay time, back to the PE's involved.

f) Executive will make use of Interval Timers (16 H/W and n S/W) to determine when next grant message can be sent.

1) If arbitration is on a cycle-by-cycle basis, then the use of interval timers will provide a regular scheduling of the "grant" messages.

2) If a process can seize a resource for an unlimited period of time, then "release" messages are needed.

g) Message contents:

Request Msg Type	PE Identification	Shared Resource ID	Priority

Grant Msg Type	SRC Msg ID	Shared Resource ID	Arbitration Delay

Release Msg Type	PE Identification	Shared Resource ID	Not Used

Figure 10.2.6-1 . Suggested contents for Message Handling

h) Internal Firmware: Consists of two control blocks:

1) Resource Control Block (RCB) contains information about each shared resource.

Address to microcode priority algorithm	
Resource service time	
Start address for set of PCB's	
No. of PCB's	Busy/Release flags

Figure 10.2.6-2 Suggested Contents for Resource Control Block (RCB)

2) Process Control Block (PCB) records the process that is waiting for a resource.

PE Identification	Priority
Accumulated Arbitration Delay	
Service/Waiting Status	
Root Flag/Link to next entry	
Address of entry in RCB	

Figure 10.2.6-3 Suggested Contents for Process Control Block (PCB)

3) Use of Control Blocks. When the Executive receives a request for a Shared Resource (SR), it finds the entry in RCB. Using data there relative to the PCB, the Executive will follow links to last entry for this SR, and will "link in" this new request within PCB.

When Executive selects an entry from PCB, the Accumulated Arbitration Delay is sent to PE along with the Grant Message.

10.2.7 Time Alignment Controller (TAC)

Purpose: The TAC controls pseudo-time for the TS.

Theory: During RTE (Run Time Executive), the user specifies the speed at which to generate the pseudo-tics. This speed regulates the pseudo-timing for the TS.

S/W: None required.

10.3 Functional Description - Segment Level

The four segments are identical (see Figure 10.0-2). Each contains a Segment n Control and BIU (as S1CBIU, S2CBIU, etc.), a Breadcast Controller, and 16 PE's. The PE description is in Section 10.4, but the other two functions follow.

10.3.1 Segment n Control and BIU (SnCBIU)

Purpose: This functional control block will coordinate with the Segment Broadcast Controller (BC). On outgoing messages, this unit will hold the message and handle the protocol until a successful send is accomplished. On incoming messages, it will screen all messages on the SBS Master bus and accept only those for a component on its segment. Then it passes the message onto its segment bus.

Theory: On outgoing messages, this block accepts the message to relay, but does return an ACC (Transfer Accepted bit in SBS format) to the sending PE. When other segment is ready to receive (SmCBIU is ready to accept message), SnCBIU will send and then wait for an "ACC" message from SmCBIU. If this "ACC" message is not received (by any sender within the chain) after so many tries and a delta wait each time, the sending SnCBIU notifies FCPI (Facility Control Processor Interface) that an error exists. FCPI will halt TAC and also send the error message to FCP.

H/W: All functions are accomplished by hardware.

S/W: None required.

10.3.2 Broadcast Controller (BC)

Purpose: Each segment has its own BC. The BC controls communications to all PE's on its segment.

Theory: The BC has two "wired-OR" lines to a dedicated register within the BIU of each PE. Line BCC1 indicates to the BC that each device (register of the PE) is ready to receive a communication. If the device does not receive the communication, then BCC2 indicates to the BC that a hardware failure has occurred in one of the PE's.

H/W: All functions are accomplished by hardware.

S/W: None required.

10.4 Functional Description - PE Level

Each PE is the same (see Figure 10.0-3). The functional description of the parts follow as the next sub-sections.

10.4.1 Bus Interface Unit (BIU)

Purpose: The BIU accepts messages for the PE and formulates and sends messages from the PE.

Theory: A set of registers exists within the BIU to send and to receive messages relative to the PE. The BIU also has to send an "Accept" or a Not-accept" response to all incoming messages. The BIU also has to receive and respond, if necessary, to same type of messages when the BIU sends a message.

H/W: All control is by hardware.

S/W: None required.

10.4.2 Arbitration

Purpose: The Arbitration module interfaces Local Memory, the BIU, and the internal bus of the PE. It decides who gets to use the PE's bus.

H/W: All is done by hardware.

S/W: None required.

10.4.3 Memory Interface Unit (MIU)/Memory Interface (MI)

Purpose: The MIU will contain those tables which allow the PE to access the instructions/data of the TS.

Theory: A Memory Map Table is defined during SGL when the user defines the memory of his TS. During RTE, when the user defines the memory locations of variables, of memory blocks, etc., relative to DEBUG/PMS, this table is further refined. Then when executing the TS, this table is always utilized to locate the address relative to TS and to physical location within PE's. This allows the monitoring of DEBUG/PMS without any additional overhead.

The Memory Map Table is in a memory of 64-bits X 2K words. The format is:

(32 bits)	(32 bits)				
BEGIN ADDR	END ADDR				
Relocation	L/S	PMS	I/O	Shared No.	TAC DECR

Figure 10.4.3 Memory Map Table Format

BEGIN ADDR - TS beginning address

END ADDR - TS ending address

Relocation - relocation to adjust for 8, 16, ..., 64-bit words; defines PE absolute address

L/S - shared or local memory block

PMS - 16 bits to define PMS event flags.

I/O	-	Memory mapped Input/Output
Shared No.	-	Shared Block Number to be sent to SRC to specify the shared block to access.
TAC DECR	-	Access time of this emulated memory block.

Since two words are used to define an entry, one thousand blocks of memory can be defined. During SGL, when the user defines memory blocks, these definitions are stored downward. During RTE, when the user defines memory and variables to be monitored, these are separated from the initial definition area and are identified with flags.

S/W: No programs are needed, but the table is defined and controlled during SGL and RTE.

10.4.4 Local Memory (LM)

Purpose: Local Memory is 16-bits X 64K words. It is used to hold the absolute program of the TS. Local Memory is filled during the execution of SGL and/or RTS, depending upon the general nature of the TS Operating System and User Application programs.

Theory: When the user is defining his TS Operating System, the selected object code is downloaded to Local Memory. When he chooses the application programs to execute under that Operating System, these programs are also stored in Local Memory. The downloading of these object programs are through FCPI (Facility Control Processor Interface). Then when the TS is run, the instructions are pulled from Local Memory by the IEC (Instruction Emulation Code), which is a microcode program in Control Store.

H/W: Hardware (data multiplexer network) is needed to divide LM into modules of 8-bits wide to store varying word lengths.

Word length	Number of modules	Number of word/mod
8	1	128K
16	2	64K
32	4	32K

Also need hardware to select correct data, and to output this data.

S/W: No software required.

10.4.5 Instruction Execution Unit (IEU/IEU & IOC)

Purpose: The Instruction Execution Unit handles the emulation of the target system. It will handle the fetch, decode, and execution of the TS Software. If needed, it will provide I/O emulation support. It also supports the time alignment, PMS, and DEBUG.

Theory: The IEU consists of the following:

a. Control Store (CS) - This is 80-bits X 4K words for containing the Executive (PEXE), the emulated instruction set (IEC), and the emulated hardware modules, all in microcode.

b. Internal Memory (IM) - This is 32-bits X 4K words for storage of data areas, TS registers, PEXE registers, etc. The 32-bits are sub-divided into 16-bits for data, and the right most 16-bits for DEBUG/PMS Flags. Also stored here are the needed software Interval Timers.

c. Registers and Arithmetic Logic Unit (RALU) - The RALU consists of 32 16-bit registers and a 16-bit ALU. The intended use for the registers is temporary storage or workspace for PEXE and the emulation. A standard partition should be set up by software to logically divide this worksapce into contexts of PEXE, the instruction emulation, I/O emulation, etc. In addition, there is no hardware restriction as to using these registers to hold the TS register data. It is a choice that is ultimately made by the emulation writer. However, the DEBUG/PMS hardware will no longer be able to cause the "automatic" collections. The user would be taxed to check PMS for himself.

The ALU portion of the RALU is capable of performing 16 simple arithmetic and/or logical instructions such as addition, subtraction, AND, OR, etc. It is also capable of shifting the results left or right, arithmetically or logically, through or around the carry, or rotating the bits on the same cycle as the arithmetic operation. Specifically there does exist a multiplication instruction which is executed on one cycle as well.

Two additional external special-purpose registers exist. One stores the condition codes created by the RALU. The other is another status register which represents any user defined status.

The ALU Condition Codes are:



- Sign for 8-bits
- Sign for 16-bits
- Integer Overflow on 8-bits
- Integer Overflow on 16-bits
- Carry 4 (from 4th bit into 5th)
- Carry 8 (from 8th bit into 9th)
- Carry 16 (from 16th bit into 17th)
- Parity on 8-bits
- Parity on 16-bits
- Zero on 8-bits
- Zero on 16-bits

d. Time alignment is accomplished in the Pseudo-time Accumulator. The user microcode for the emulation of the instruction set will load the execution time of each macro instruction into Register 3 of that module.

e. Interval Timers

- Will have 16 hardware Interval Timers.
- Use H/W Interval Timer Ø to reference the stack of software Interval Timers.
- Use H/W Interval Timer 15 always for pseudo-time DEBUG and PMS.
- Can have as many software Interval Timers as needed.
- Interval Timers are assigned to the emulated

I/O modules, i.e., disk unit, disk controller, etc. The emulated device handler will, as its first function, store the time required for this emulation step. When this time has elapsed, control will go to PEXE. PEXE contains an interrupt handler to process the condition that more time is required to emulate the emulation process.

f. Priorities

- A Priority Table needs to be established within PEXE to handle all MMS interrupts. This table will also contain the required addresses into PEXE for the different Interrupt Handlers.

H/W: 1. Control Store and Internal Memory

2. Arithmetic Logic Unit

3. Hardware Interval Timers

4. Related circuitry to Pseudo-time Accumulator, to handle interrupts, etc.

S/W: 1. PE Executive

2. Emulated Instruction Code

3. Various hardware emulators

10.4.6 Control Store (CS)

Purpose: Control Store is a memory block within IEU (Instruction Execution Unit) which contains the microcode modules of PEXE, IEC for a TS, and all the emulated hardware modules for the TS. Its size is 80-bits X 4K words.

Theory: The microcode instructions written to define a designated TS are stored within Control Store. These are utilized by IEU to execute the TS macro-code.

H/W: None required.

S/W: a) PEXE is a vendor -supplied microcode PE Executive which is independent of the selection of TS. It contains the mapping (linking) tables to allow the execution of any TS instruction. It responds to general type interrupts and interfaces to the master modules of MMS to handle I/O.

b) IEC (Instruction Emulation Code) is a microcode module to interface TS macro instructions to the PE microcode. A sample IEC is vendor-supplied, and a manual describing how to write this module is also vendor-supplied. IEC's for additional TS's are then user-supplied.

c) Emulated H/W modules are microcode modules to emulate the actions of various hardware units/peripherals of a TS. Again, several types are vendor-supplied to emulate the parts of a selected TS. Also, a manual will be supplied to instruct others as to format and ways to write modules for future TS's. The depth of these emulations are relative to the needs of the user.

BITS	
4	Sequencer Op Field
16	Displacement Field A. Next Address (Micro) B. ALU Cond. Code Select C. Loading of Register 1. ALU Cond. Codes 2. Loadable Cond. Codes 3. Mask Word for DECODE D. ALU Reg. Select Field
16	Pseudo-time Dec. Field
8	ALU Reg Select (Source/Destination)
10-16	IEU Addr. Select (M-INST Type)
1	ALU Carry In
4	ALU Function Select
3	ALU Shift MUX Select
4	Decode Logic Select
8-16	Individual Control Lines A. Pseudo-time Decrement B. MI Start Signal C. MI Incr. Signal D. Memory Byte/Word Select E. BIU Clock Signal F. BIU Reset Signal
7	Condition Code Input Select A. 16 ALU Condition Codes B. 16 Loadable Condition Codes C. 32 PMS Conditions D. 16 Interval Timer Interrupts E. 5 MI Control Codes 1. Complete and OK 2. Error 3. PMS Block 4. SRC Action 5. I/O Address F. 2 BIU Codes 1. Service Request 2. Buffer Full G. Combinations

Figure 10.4.6 Proposed Format for Micrccode Word

10.4.7 Internal Memory (IM)

Purpose: Internal Memory is a memory block within IEU (Instruction Execution Unit) which contains the data words required by the microcode modules within Control Store. The size of this memory is 32-bits X 4K words.

Theory: The data needed by the microcode programs within Control Store are located in Internal Memory. These data include variables, registers, tables, stacks, etc. The 32-bit words contain 16-bits of data on the left-most half and 16-bits of DEBUG/PMS flags on the right-most half.

H/W: None required.

S/W: Defined as needed within PEXE, IEC, and emulated H/W modules.

10.4.8 Pseudo-time Accumulator

Purpose: The Pseudo-time Accumulator is a series of registers and associated logic to control pseudo-time for the PE.

Theory: During RTE, the user specifies the time value of a pseudo-tic and the value of the pseudo-time window. These values are stored in the Pseudo-time Accumulator (PTA) of each PE. Then, as each macro-instruction of the TS begins execution, its required execution time is sent to the PTA. During execution of the macro-instruction, pseudo-tics are sent by the master TAC to PTA. The PTA maintains a differential time window of the amount of time used by the PE and the amount issued by TAC. If the PE moves ahead of the window, the PTA informs the PE to go into an idle state. If the PE lags behind the window, the PTA informs the TAC not to issue any more tics until the PE advances back into the window.

H/W: All is accomplished by H/W.

S/W: None required.

10.5 Abbreviations Used in Section 10

ACC	Transfer Accepted Bit
ALU	Arithmetic Logic Unit
BC	Broadcast Controller
BCC	Broadcast Control Line
BCC1	Broadcast Controller Line 1
BCC2	Broadcast Controller Line 2
BIU	Bus Interface Unit
CRT	Cathode Ray Tube
CS	Control Store
DEC	Digital Equipment Corporation
DMA	Direct Memory Access
FCP	Facility Control Processor
FCPI	Facility Control Processor Interface
H/W	Hardware
IEC	Instruction Emulation Code
IEU	Instruction Execution Unit
IEU (&IOC)	IEU with Function of IOC (Input/Output Controller) included
IM	Internal Memory
IOP	Input/Output Processor
IOPT	Input/Output Port Table
LM	Local Memory
MBC	Master Broadcast Controller
MI	Memory Interface Unit
MIU	Memory Interface Unit
MMS	Multiple Microprocessor System

MUX	Multiplexer
PCB	Processor Control Block
PE	Processing Element
PEXE	PE Executive
PMS	Performance Monitoring System
PMSEX	PMSI Executive
PMSI	PMS (Performance Monitoring System) Interface
PTA	Pseudo-time Accumulator
RCB	Resource Control Block
RTE	Run-time Executive
SBS	Synchronous Busing Structure
SGL	System Generation Loader
SmCBIU	Segment m Control and Bus Interface Unit
SnCBIU	Segment n Control and Bus Interface Unit
SR	Shared Resource
SRC	Shared Resource Controller
S5BC	Segment 5 Bus Controller
S/W	Software
TAC	Time Alignment Controller
TS	Target System

11.0 OVERVIEW - MMS SOFTWARE DESIGN

The software design of MMS follows one approach to top-down methodology. Top-down design is a systematic organization of known facts and known objectives into a cohesive, graphic system. The top level provides the overview. Then each lower level describes the functional steps ("children") to accomplish its "father" level.

The graphic approach utilized within this document is designed after that set forth by SofTech, Inc.* Figure 11.0-1 is that company's "Structured Analysis Model". Since the control flow of charts is always from left to right, and from top to bottom, the modified SA charts used within this document are easily understood. On the left-side of all these boxes will be the input data coming into the box. The right-side will identify the output data and the output control flow. Inside a box will be identified an activity, which will be identified as an acting verb and the object of that action.

Input into the top of the box will indicate the needed control data, as input tables, etc. In most cases, the linking between the boxes or nodes of a chart will be into the top of the box to indicate implied connecting tables to achieve that next step.

Input into the bottom of these boxes will identify the mechanism or processor needed to accomplish the specified action named within the box.

Figure 11.0-2 shows the top level design of MMS. The user, in order to utilize the MMS configuration, will have much

*See "Structured Analysis Reader Guide", 9022-78.2, May, 1978, by SofTech, Inc., 460 Totten Pond Road, Waltham, MA 02154

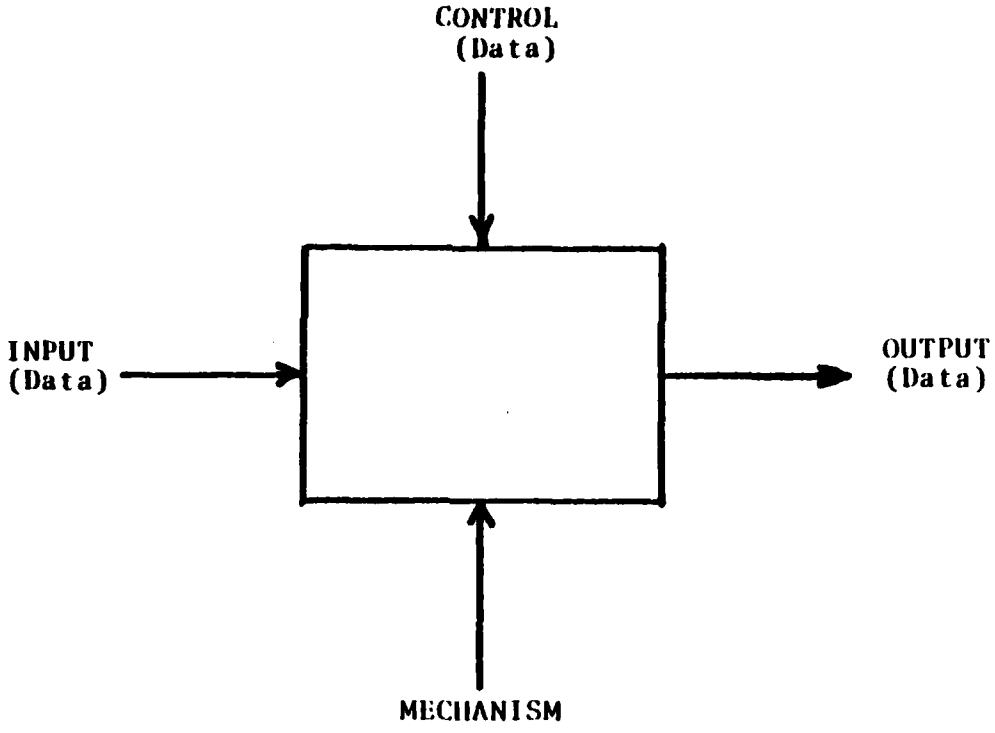


Figure 11.0-1 Structured Analysis Module (SA)

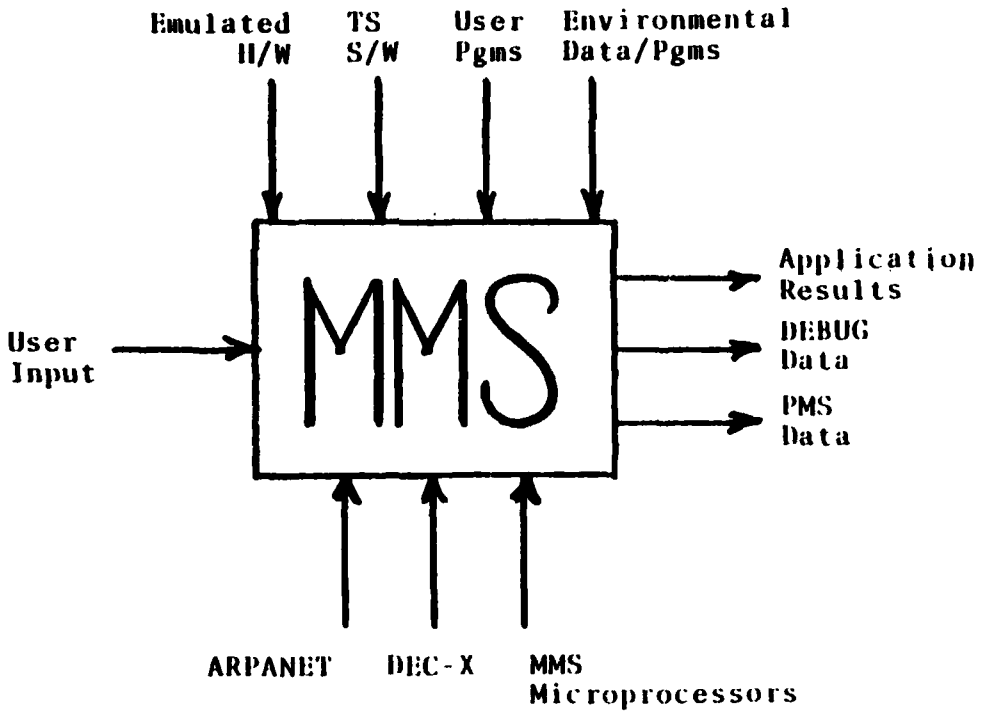


Figure 11.0-2 Top Level Design of MMS

influence on the total operation of MMS; thus, the indication of "User Input" on the left side. On the right side are three arrows to depict total output which the user can obtain through the execution of MMS: any output from the user's application programs, any DEBUG results when checking program execution, and any PMS data collected on the program execution.

The types or categories of files available to the MMS user are four in number:

1. Emulated H/W, which includes the instruction set and all hardware modules to emulate the total target system.
2. TS S/W, which will include the operating systems.
3. User Pgms, which includes the application programs that the user wishes to execute under some operating system on the configured Target System.
4. Environmental Data/Pgms, which includes the data files and/or programs which constitute the environment and provide input to MMS.

Figure 11.0-3 shows the modular design of MMS. This is a "busy" chart, but necessary modules or programs to fulfill the mission of MMS are shown here graphically to illustrate their relationships.

The microcode programmer can use either an "E-Code Compiler" or the "Microcode Assembler" to generate the file of "Emulated H/W Modules." A "Target System Configuration" is designed to be the beginning this file of "Emulated H/W Modules". "PEXE" is an executive to be down-loaded to every PE; PEXE is vendor supplied to handle interrupts and standard I/O processes of the parts of TS.

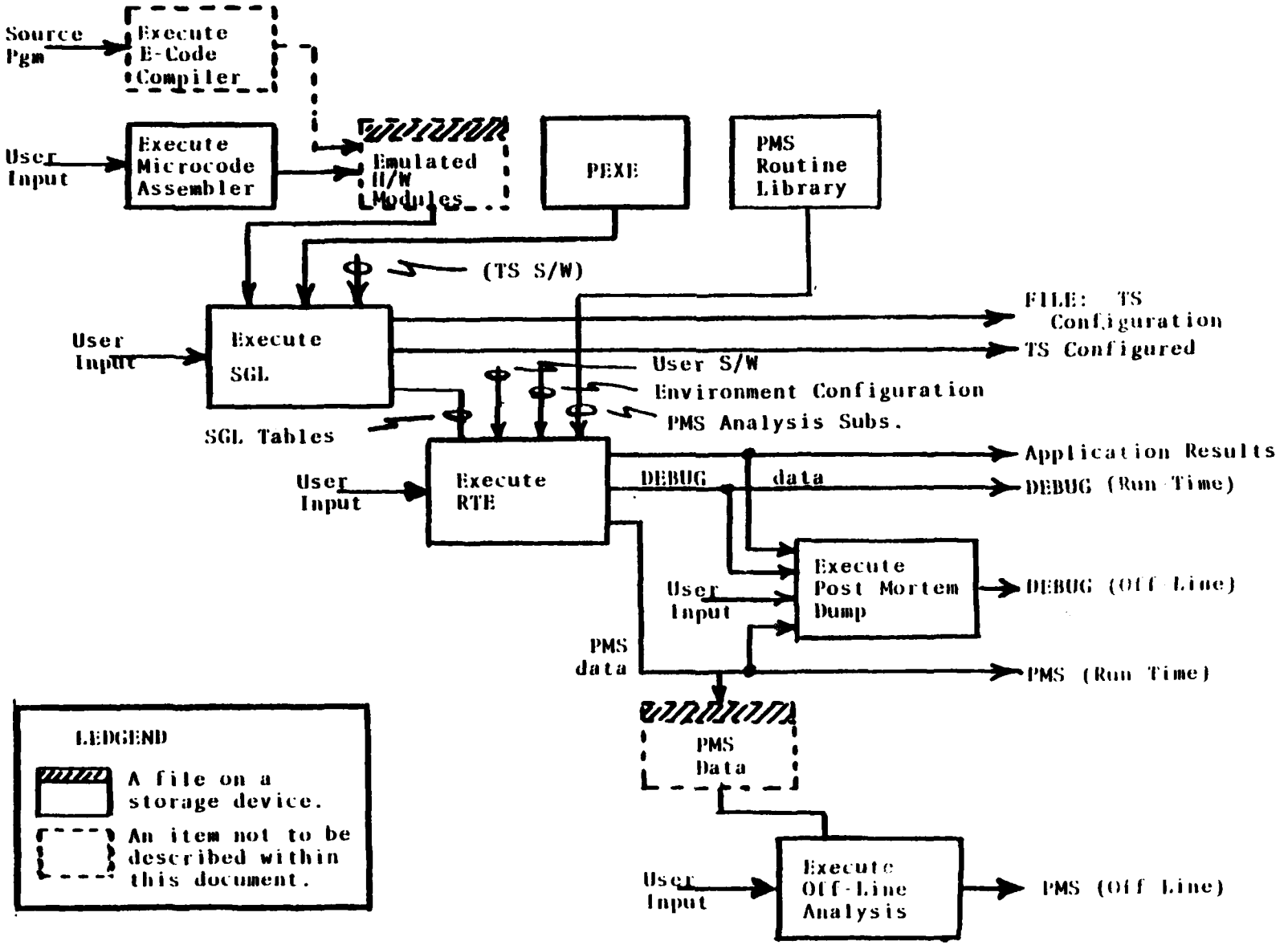


Figure 11.0-3 Detailed Modular Design of MMS

The MMS User calls "SGL (System Generation Loader)" to help him to select the parts of his TS, and then SGL handles linkages. The MMS User then calls "RTE (Run-Time Executive)" to finish the configuration of the TS software and to handle execution of the TS programs under DEBUG/PMS.

The user can call "Post Mortem Dump" program to check the static conditions of any part of the TS or of the FCP (Facility Control Processor). This program can be called whether or not MMS was operated under DEBUG/PMS options.

If the user has a large amount of data collected on the TS execution, this data has been stored within a file, probably on disk. The user can select which routines he wishes to utilize for the analysis and for the graphical display of that data.

Another package required as part of MMS is one necessary to test the hardware components of MMS. The FCP computer will have its set of diagnostic routines. These will be exercised, also.

The nine modules thus specified will be the next nine sections. Each will be described verbally. Then a "Node Tree Diagram" will specify all charts necessary to graphically describe the functional steps of that module; i.e., will illustrate the family tree of diagrams. Each of these charts will be composed of two to six pieces (boxes or nodes).

The design of MMS software has fulfilled the purpose of being modular. This modularity has been carried down to the lowest level. Programming and testing should easily follow this design standard to aid modularity of programming and ease of documentation.

As a review, the following software packages are part of the MMS Software Design package:

- E-Code Compiler (Section 13)
- Microcode Assembler (Section 14)
- PEXE (Section 15)
- Diagnostic Programs (MMS Hardware)(Section 12)
- SGL (Section 16)
- RTE (Section 17)
- Post Mortem Dump (Section 18)
- PMS Off-line Analysis Routines (Section 19)
- A Target System (Section 20)
- Executives for IOP, PMSI, and SRC of the "Big 7" (Section 23.2 - 23.4)

11.1 Abbreviations Used in Section 11

DEC-X	Digital Equipment Corporation computer not yet specified
E-Code	Emulation-Code Language
FCP	Facility Control Processor
H/W	Hardware
IOP	Input/Output Processor
MMS	Multiple Microprocessor System
PEXE	PE Executive
Pgms	Programs
PMS	Performance Monitoring System
PMSI	PMS (Performance Monitoring System) Interface
RTE	Run-time Executive
SA	Structured Analysis
SGL	System Generation Loader
SRC	Shared Resource Controller
Subs	Subroutines
S/W	Software
TS	Target System

12.0 MMS DIAGNOSTIC PROGRAMS

The purpose of the MMS Diagnostic Programs is to isolate hardware faults down to the functional level. This package has to be a stand-alone unit that can test all components of MMS when it is necessary to do hardware maintenance. The same routines will be used by SGL (System Generation Loader) to determine which components of MMS are operational.

The approach required of the diagnostic package is that all tests on the PE level store the results in that PE. A test will set a specified bit within a word for a "good" test and another bit for a "no good" test. This table of results will be uploaded to the FCP as part of the total diagnostic package. Similarly, the modules on the Master Segment will be tested, results stored, and then uploaded to the FCP.

The stand-alone diagnostic package will have the Master driver reside in the FCP. "Slave"driver programs will be down-loaded to the PE's and to the necessary modules of the "Big 7". The Master in the FCP will send start signals to those programs within the PE's and the master modules which are part of the "Big 7". After this initiation, the master program in the FCP will collect results from the others. Figure 12.0-1 shows what programs are included within this design.

The diagnostics package will consist of a main program which will call subroutines within the FCP. The routines within the FPC will comprise the "Master" portion of the tests. Figure 12.0-2 illustrates this approach, and will also illustrate the descriptive discussion which follows.

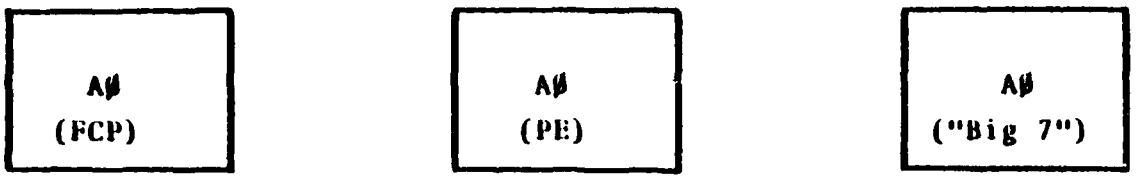


Figure 12.0-1 Top-Down Design Chart

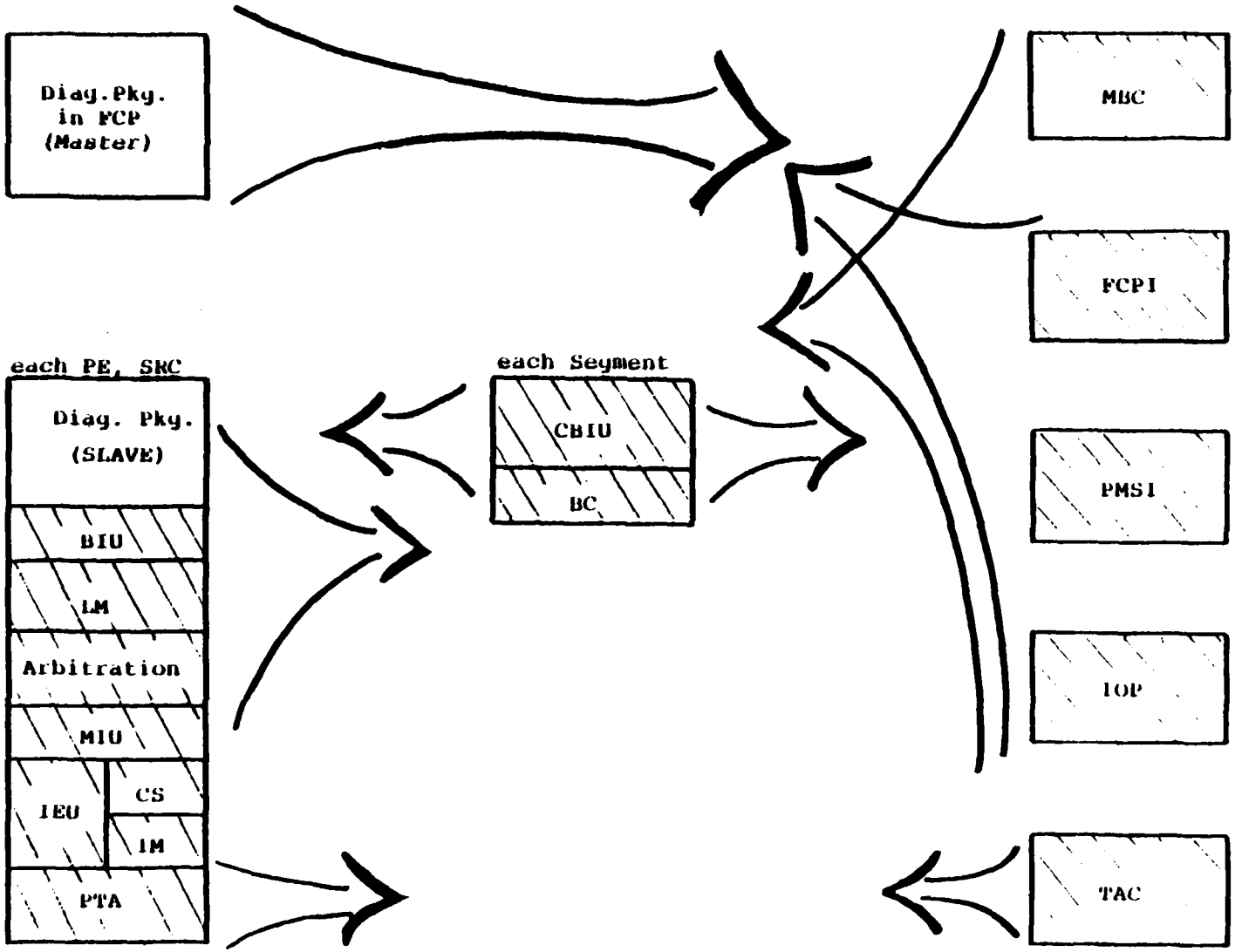


Figure 12.0 2 Diagnostic Package Interactions
(Slashed blocks identify hardware modules)

Section 10, "MMS FUNCTIONAL DESCRIPTION", discussed the hardware modules of MMS and what the functions of each are. The Master Segment is comprised of the "Big 7":

MBC - Master Broadcast Controller

FCPI - FCP Interface

PMSI - PMS Interface

IOP - I/O Processor

TAC - Time Alignment Controller

SRC - Shared Resource Controller

S5BC - Segment 5 Bus Controller

Section 10 identified the functions of each of these hardware components. All tables, registers, memories, etc., of these modules need to be massaged. Tests have to be written to test the functional purpose of each module, also. Thus, the IOP has to be able to download data/programs to the PE's; it has to be able to upload blocks of data from a PE to the FCP. The PMSI has to be able to accept PMS data from a PE, to assess the PMS function type, and then to select one of several control paths. All these tests have to verify internal actions and have to verify the interactions with the PE's. Once all diagnostics within each module of the "Big 7" have been processed, the results are to be uploaded to the FCP.

Since each segment is a step lower than the parts of the Master Segment, each of these has to be tested. The passing of messages by the segment BC (Broadcast Controller) and the segment CBIU (Control and BIU) between the Master Segment and the

individual PE's will constitute a functional test. Then the additional required tests to identify board functions will be added later.

The next lower set of tests will be that needed to verify that each PE is operational. A diagnostic package will be downloaded to each PE. The "Master" in the FCP will then initiate the execution of these "slave" programs. The parts of the PE have also been identified in Section 10.

The diagnostic program downloaded to Control Store of the PE will have to verify that all parts of the PE are operational. The parts identified in Figure 12.0-2 are:

BIU - Bus Interface Unit

LM - Local Memory

Arbitration

MIU - Memory Interface Unit

IEU - Instruction Execution Unit

CS - Control Store

IM - Internal Memory

PTA-Pseudo-Time Accumulator

Since each PE has a word length of 32 bits within Internal Memory, each word can reflect the results for 16 tests. Each test will have control over the setting of 2 bits: one for "good" and one for "no good". Some tests are to be run several times. Thus, a table is required for the storage of results. The uploading of this table from each PE to the FCP through the FCPI will constitute part of the functional test of FCPI.

The functional tests could be identified as:

A. Master Broadcast Controller (MBC)

Purpose: to broadcast 1-word messages to all PE's

Test Pgm: FCP sends a message to MBC to broadcast to all PE's. When PE receives this message, it will fill word in Test Table that reflects MBC Test Complete.

Number of Trials: suggest that at least 4-8 tests be executed.

B. Master Broadcast Controller (MBC)

Test Pgm: one PE at random is selected to send a broadcast message to MBC. Again, each PE will reflect that received message in Test Table.

Number of Trials: suggest that at least 4-8 tests be executed.

C. Facilities Control Interface (FCPI)

Purpose: to download or upload large blocks of data or programs.

Test Program: FCP has the FCPI down-load the test programs for each PE. The PE can use the FCPI to up-load the results. (Have PE also use IOP to up-load the control words of test results.)

Number of Trials: one down-load from FCP to each PE should be sufficient.

Note: The FCPI is designed to verify data. One PE should also request this function. Two tests should be used:

a. Verify that one block of data sent twice does check.

b. Change n words for one set of data. See if n errors occur.

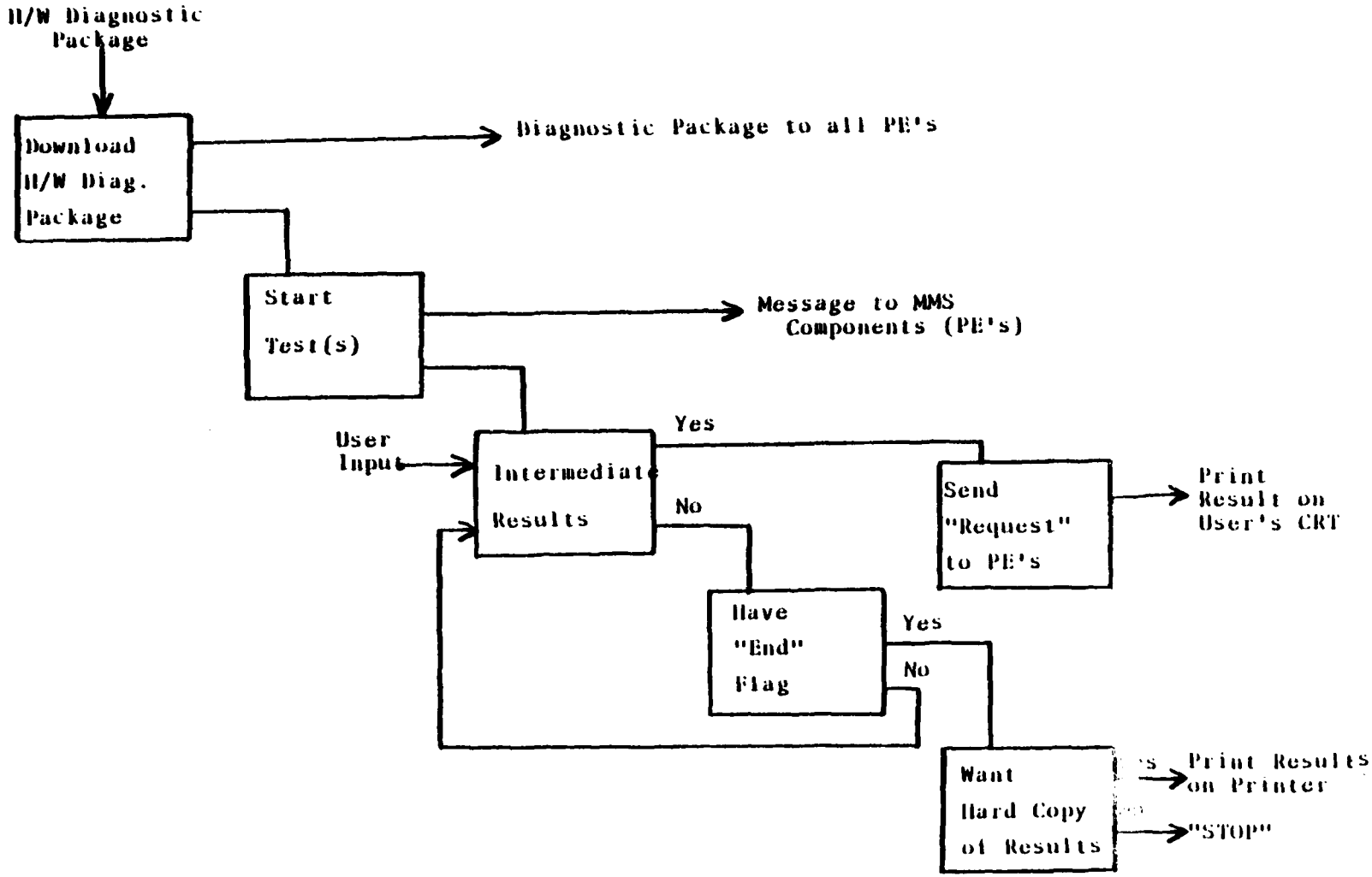
Initially, the approach was that the MMS diagnostic package would be user selectable. A menu of available types of tests was to have been listed on a CRT, and then the user would have selected those in which he was interested. Bus speeds have been studied, and the MMS structure has been evaluated relative to the diagnostic requirements. The best approach is to have all tests run without any interaction from the user. More positive results can be obtained, plus the time required for this total package is felt to be about 1 to 3 minutes.

AØ - HARDWARE DIAGNOSTICS (FROM FCP VIEW)

The diagnostic package will have the "Master"module reside in FCP. Separate "Slave" modules will be downloaded to all PE's. Each number of the "Big 7" will have its own individual "Slave" test package. The Master initiates execution of all the Slaves.

Then the Master waits either for an interrupt from the user or an interrupt from one of the slaves. If the user interrupts to request intermediate printouts on his CRT, the Master sends the request to the members of MMS. When the Master receives the answers, he prints the results.

If the user wants a hard copy printout at the end of the diagnostic run, the Master will receive all tables generated by MMS. These tables will be DMA'd to the memory of FCP. The Master will interpret all the tables and print a summary for the user.



A∅ (ECP)

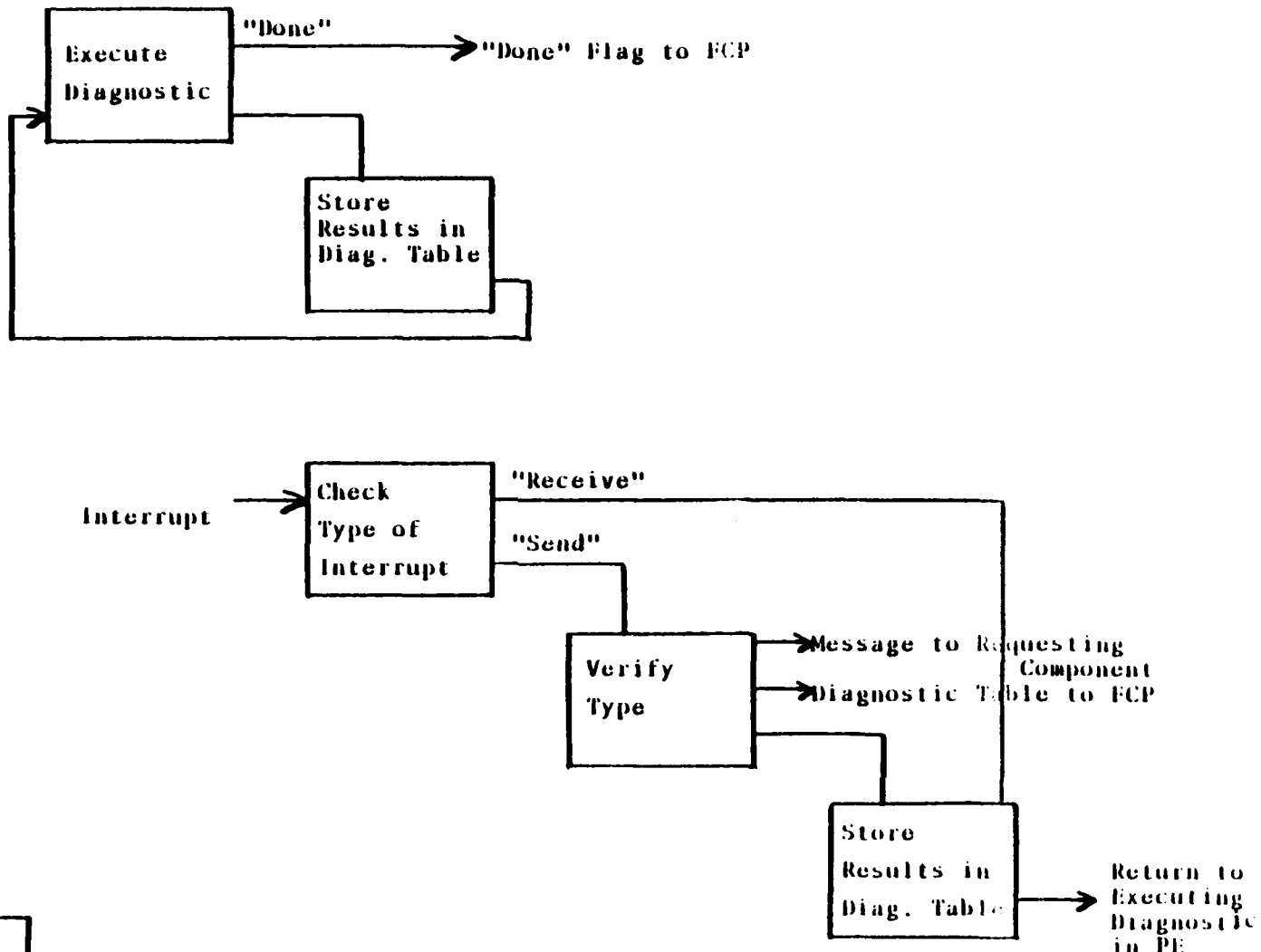
H/W Diagnostics

AØ - HARDWARE DIAGNOSTICS (FROM PE VIEW)

Each PE will have a diagnostic program downloaded to it from the FCP. FCP will give the start message to the TAC, which will start sending pseudo-tics to all PE's.

Each PE will test all of its internal components and will store the results within a Diagnostic Table. When completed, the PE will then send a completion flag to FCP. FCP may request the results, and will tell each PE when it is ready to receive the results.

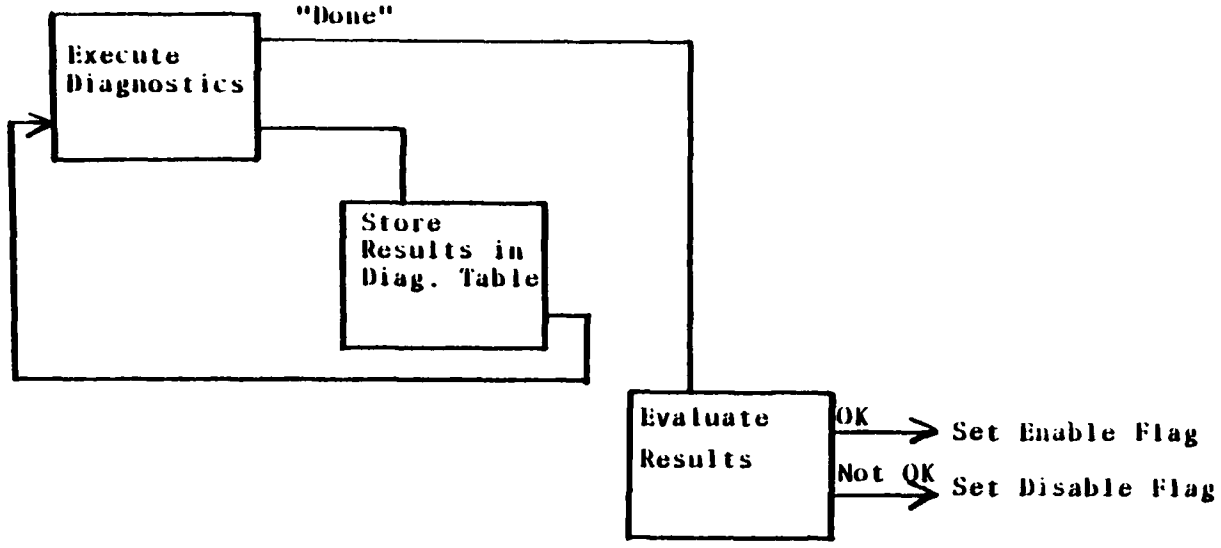
Within the PE, whenever an interrupt is received, the PE will check to see what type of interrupt - PE to do something or PE to receive results for storing. If the PE is to respond with an action, it will send what is requested by FCP or by another component of MMS.

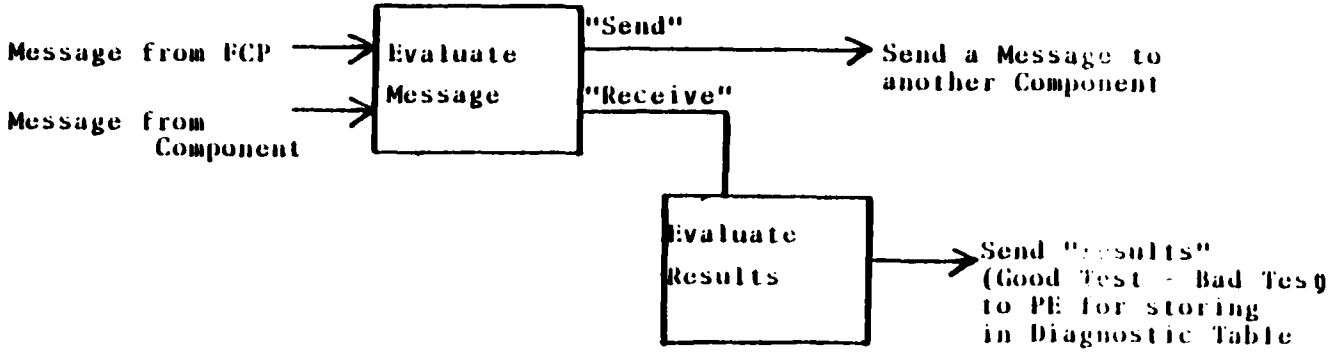

Execute Diagnostic
"Done"
"Done" Flag to FCP
Store Results in Diag. Table
Interrupt
Check Type of Interrupt
"Receive"
"Send"
Verify Type
Message to Requesting Component
Diagnostic Table to FCP
Store Results in Diag. Table
Return to Executing Diagnostic in PE

AØ - HARDWARE DIAGNOSTICS (FROM "BIG 7" VIEW)

Once these seven modules are designed modularly it is possible to define these tests. Where possible, each module should have tests similar to those for the PE's - one which will check all registers, memories, etc.

Then, the MMS-defined functions should be evaluated for each module. When the above tests are done and the enable flag is set, each module will accept data from one or more PE's which will help simulate the defined functions. Then several PE's will interact through these modules on the Master Segment. If needed, the data received under simulation conditions is evaluated and the results are sent to a PE for storage in its Diagnostic Table. In other cases, a message is sent to one of the PE's.





AØ ("Big 7")
H/W Diagnostics

12.1 Abbreviations Used in Section 12

BC	Broadcast Controller
BIU	Bus Interface Unit
CBIU	Control and BIU
CRT	Cathode Ray Tube
CS	Control Store
Diag.	Diagnostic
Diag. Pkg.	Diagnostic Package
DMA	Direct Memory Access
FCP	Facility Control Processor
FCPI	Facility Control Processor Interface
H/W	Hardware
IEU	Instruction Execution Unit
IM	Internal Memory
IOP	Input/Output Processor
LM	Local Memory
MBC	Master Broadcast Controller
MIU	Memory Interface Unit
MMS	Multiple Microprocessor System
PE	Processing Element
Pkg.	Package
PMS	Performance Monitoring System
PMSI	PMS (Performance Monitoring System) Interface
PTA	Pseudo-time Accumulator
SGL	System Generation Loader
SRC	Shared Resource Controller
S5BC	Segment 5 Bus Controller
TAC	Time Alignment Controller

13.0 E-CODE COMPILER

This higher-order language (HOL) will significantly simplify the task of building emulators. The language should be one with which the users can be comfortable. It enables the user to write instructions such as: "FETCH MACRO", "DECODE BIT 3, 7", etc. The output of this compiler will be directly executed by the PE's within MMS. Thus, this E-Code Compiler is used in place of the Microcode Assembler.

The advantages for incorporating the E-Code Compiler into MMS rather than the microcode assembler are many. Less program statements will be required by the user to incorporate the same emulations. The user will be able to accomplish more emulations also because of ease of use of the language. Error-free coding will result sooner. A higher-level language is easier to understand, to update, and to document. A well-defined E-Code language will eliminate the need for the microcode assembler. Thus, better implementation of MMS will save the user time and money.

The E-Code Compiler could be implemented in MMS, as well as having the microcode assembler. But, additional money could be saved by implementing this E-Code Compiler and not including the microcode assembler.

The specification of the HOL language is not part of this study, but it is imperative that this language be defined before the building of MMS. The impact of substituting the E-Code Compiler for the Microcode Assembler should be ascertained.

13.1 Abbreviations Used In Section 13

E-Code	Emulation Code
HOL	Higher-Order Language (also HLL for Higher-Level Language)
MMS	Multiple Microprocessor System
PE	Processing Element

14.0 MICROCODE ASSEMBLER

The Microcode Assembler is a vendor-supplied software package that resides on the FCP. The purpose of this package is to enable the programmer to design the hardware emulation modules and the module to emulate the target computer instruction set.

The programmer using the Microcode Assembler would have to have knowledge of:

- the language set for the Microcode Assembler.
- the language set(s) of the assembler(s) for the Target System computer(s).
- the emulation requirements of the TS, in order to do emulations to the required detail.
- the hardware details in order to design emulation modules.
- MMS in order to design the configuration of target systems within the MMS structure.

The microcode programmer will thus have to be someone with specialized skills to enable "good" application of MMS. The general user will not be this person. The microcode programmer will generate libraries of emulation modules for various computer systems. The user will utilize these libraries to design networks of computer systems, or Target Systems.

The detail level of any emulated device should be only as in-depth as the purpose of the emulation. As the library of these emulators increases, a separate descriptive area should be maintained to distinguish levels of complexity. The use of

version numbers should also be helpful to identify the various levels of complexity. Note that a version number of 3 doesn't mean that it is more complex than a version number of 2; the 3 means that it was produced after version 2.

It is recommended that the emulation of the instruction set be very comprehensive and complete. It should not be necessary to have several versions of this, the IEC.

Figure 14.0-1 identifies the hierarchy of the tip-down design charts included for the Microcode Assembler. The Microcode Assembler will accomplish these four tasks:

1. Associate symbols and labels with machine addresses.

2. Produce machine code (binary) to perform the specified functions: emulate the instruction set, emulate a hardware device, etc.

3. Connect the program to the PE executive for necessary services.

4. Produce a module of Identification Section and Program Section to be accessible by SGL.

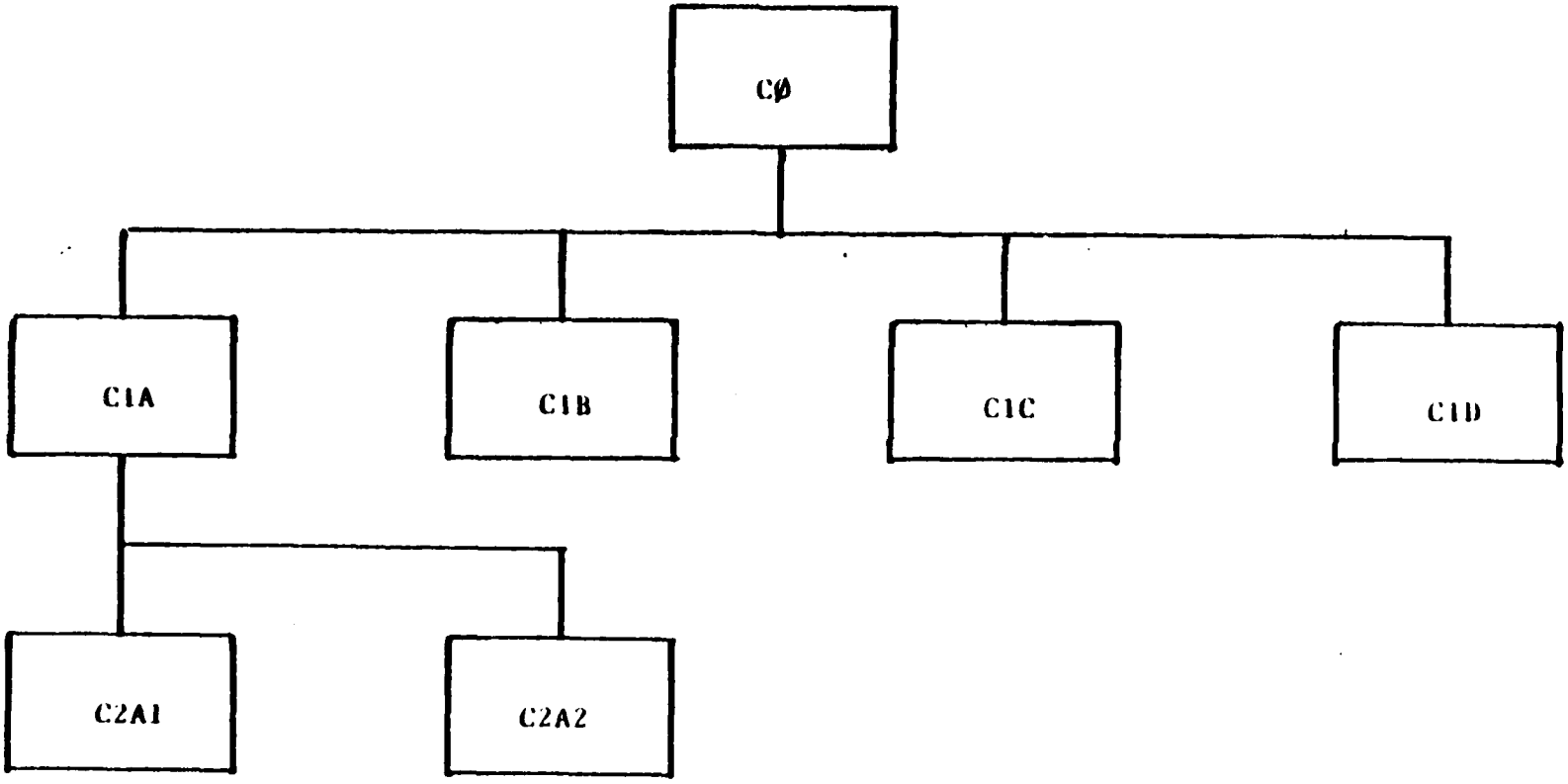


Figure 14.0-1 Top-Down Design Charts

CØ - MICROCODE ASSEMBLER

The Microcode Assembler accepts, as input, a source program in microcode language format. This source program is written as part of the Target System (TS) Emulation. The output of the assembler is either the Emulation Code in object program form or an Error list. The proposed form of the object module is given in Figure 14.0-2.

Using the proposed format illustrated in Figure 14.0-2, the object module contains several header sections which permit the user to finalize the design of the emulation module so that it fits his emulation needs. The first part of the header is the ID, which allows the selection of the modules from the libraries. The next section, "User Question Section", allows SGL to query the user as to his choice of some data definitions, speeds, etc. The "User Register Section" allows the configuration of registers of a TS computer. If the emulation is for some hardware device, etc., the next three sections allow the definition of status words, of I/O Buffers, and of I/O Ports. Because the lengths of the above areas are variable, a specified code is then utilized as the beginning of the object code. The object code module length is also variable, so another flag is defined as Ending Flag for the object module. The next variable defines the data area for the emulation module. If this emulation is for an instruction set, then the register area and data area are defined. If the emulation is for a hardware device, then timing is defined, identification of the assigned Hardware/ Software timer, some words for TS status words, some words for the buffer area, some words for I/O Port definitions, and then the area

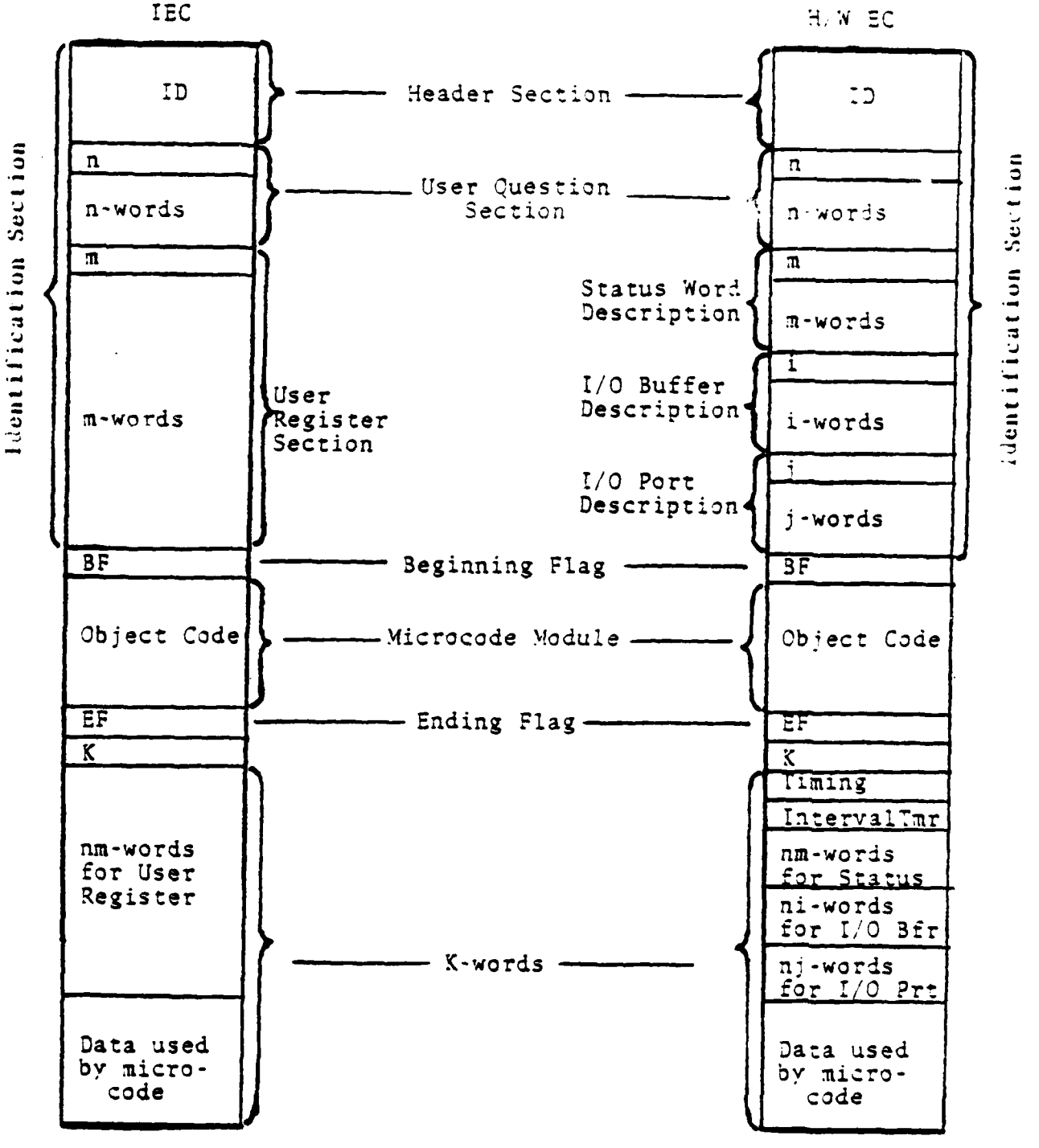


Figure 14.0-2 Proposed Format of Object Module created by Micro-assembler. Every Instruction Set or Hardware Emulation Module will have one of the above formats so that SGL can create the necessary linkages between modules.

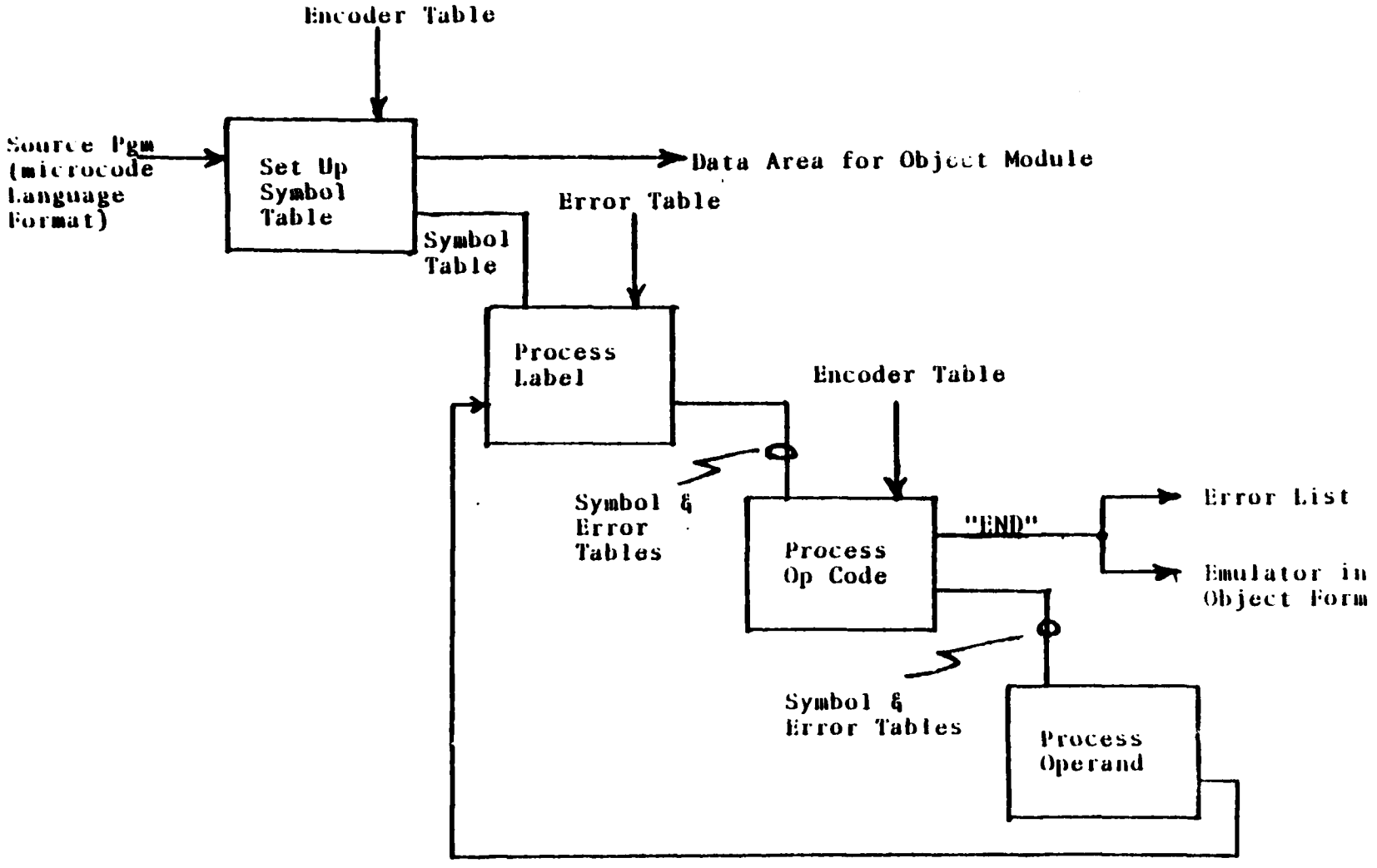
for data definition.

Figure 14.0-2 illustrates that a format has to be defined so that SGL and RTE can utilize and link these modules. Section 20, "Target System", discusses this format and its need.

The microassembler is a two-pass assembler. The first pass of the assembler processes all of the source code and builds the Symbol Table and creates the Data Area for the Object Module.

The second pass of the assembler creates the binary microcode instructions. To accomplish this task, three different processes are required. Labels and instructions need to be verified against the Symbol Table addresses to insure their correctness. The instruction needs to be decoded, with the use of the Encoder Table, into the correct number of binary microcode instructions. In addition, any operands need to be processed and placed in the appropriate binary microcode.

When an "END" statement has been encountered, processing stops. The object code or Error List is output.

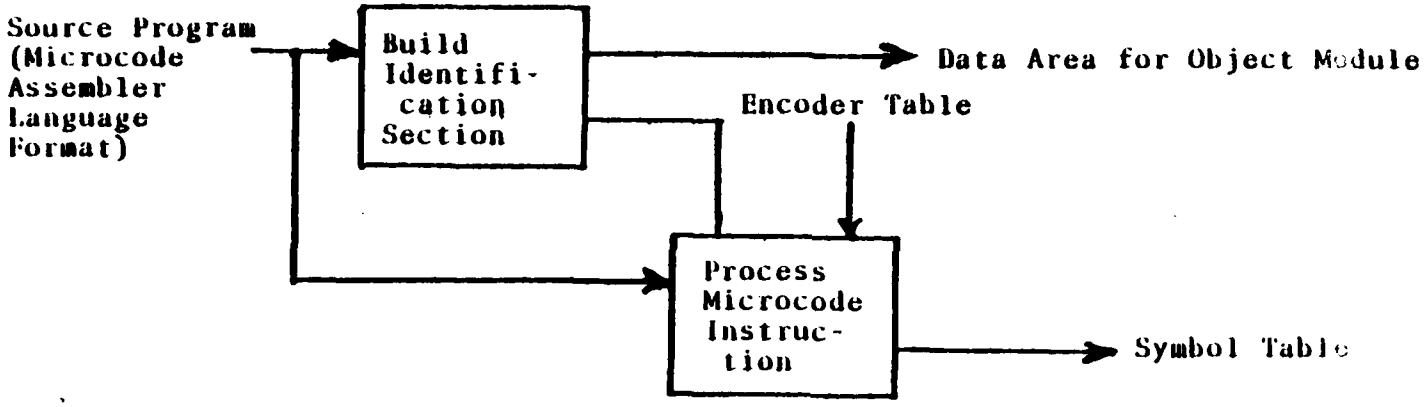

Encoder Table
Source Pgm (microcode Language Format)
Set Up Symbol Table
Data Area for Object Module
Error Table
Symbol Table
Process Label
Encoder Table
Symbol & Error Tables
Process Op Code
"END"
Error List
Emulator in Object Form
Symbol & Error Tables
Process Operand

C1A - SET UP SYMBOL TABLE

The setting up of the Symbol Table involves two distinct functions: Build Identification Section and Process Microcode Instructions. These two functions create one Symbol Table which contains all the information needed on any alpha-numeric symbol used in the source program module.

If values have to be resolved at SGL time or if linkage areas have to be defined for SGL or for RTE, these areas formulate parts of the Identification Section.

At the same time that the Symbol Table is being built, the Data Area for the Object Module is being reserved in memory. This area can then be used by SGL.



C1A Set Up Symbol Table

C2A1 - BUILD IDENTIFICATION SECTION

The building of the Identification Section (Header, Question, User Register, Status Word, I/O Buffer, I/O Port) is the first part of creating the Symbol Table.

The Header Section does not involve putting any entries in the Symbol Table. The assembler checks the name and version of the emulation and verifies that there is no other emulation with the same name and version. If there is, then there would be an error return; otherwise, the entry which identifies the EC module is placed in the area (disk file) directory.

The Question Section contains all of the user inputs that are resolved at SGL time. The inputs are identified by variable names used in the EC itself. The names are stored in the Symbol Table as external values. A memory word(s) in Internal Memory is assigned to each variable. The user is given the minimum and maximum values (if applicable) and the default value for each variable. If the user specifies a value other than the default value, the new value is stored at the address indicated for it in the Question Section.

The Status Word, I/O Buffer, and I/O Port Sections pertain to the H/W EC module, and the User Register Section pertains to the IEC module.

The Status Word Section defines all of the Status Words used by the H/W EC's. In addition, the breakdown of the Status Word(s) is given in this Section. This breakdown is needed to inform SGL of the format of the Status Word(s). Each Status Word is given a memory word(s), and has its name and address

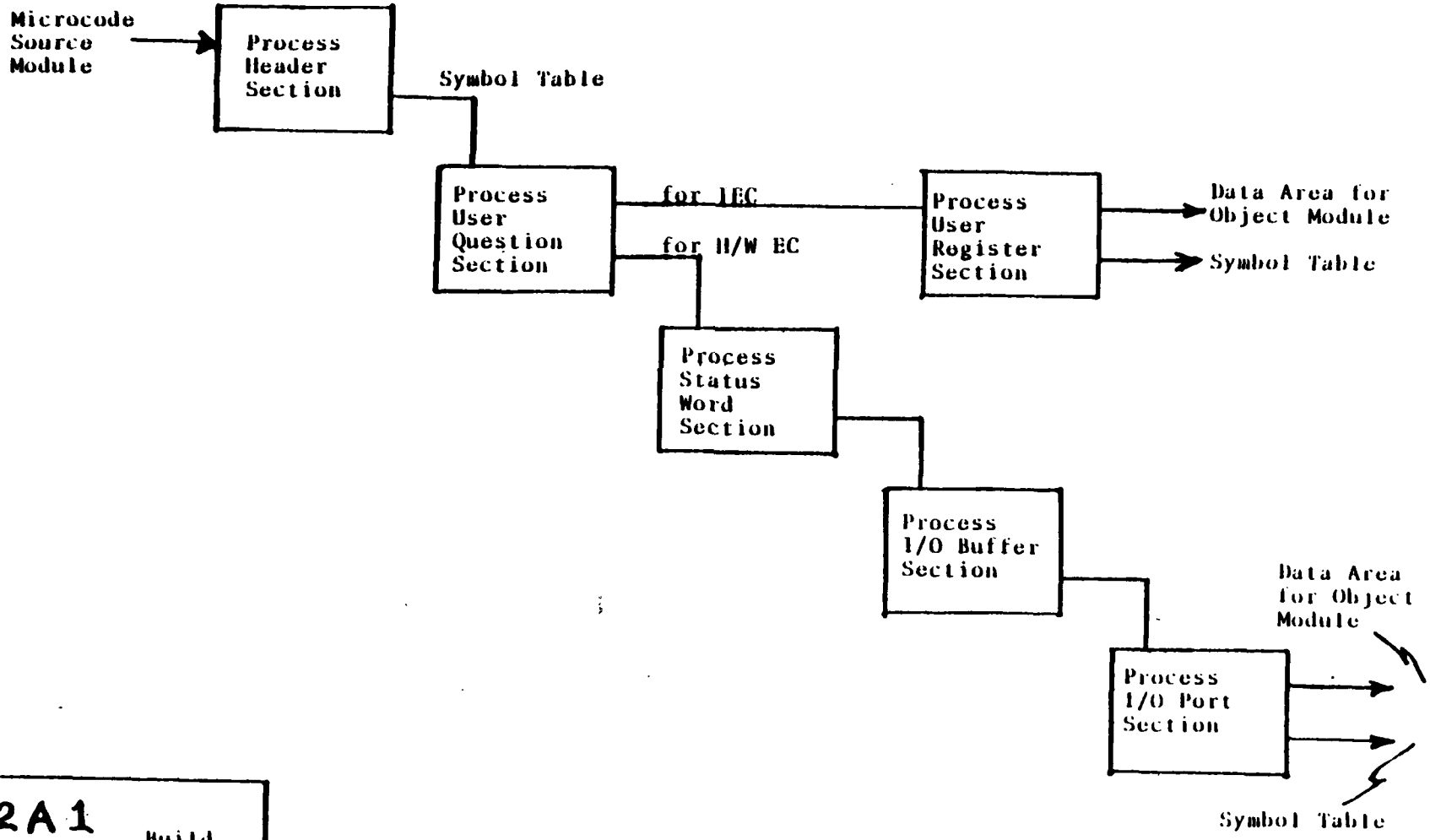
placed in the Symbol Table for use in resolving addressing in Pass Two of the assembler.

The I/O Buffer and I/O Port Sections are handled similar to the Status Word Section. The Symbol Table reflects the memory reserved for each of the types of words.

Also for the modules of H/W EC's, a place for Timing and for the address or number of the Interval Timer is reserved in memory.

See Figure 14.0-2 for the format relative to all emulated modules.

The User Register Section contains all of the registers that are used in the IEC module. There are two purposes for identifying these registers. The first one is that the micro-assembler needs to know information about the registers to resolve the addressing and sizing of the registers. The second purpose is to allow RTE to have this information so that, during PMS, register values can be accessed. The information in this section, which is identified by the microcode programmer, is merely the register bit value in the IEC. During assembly, the User Register Section is updated with the actual hardware address of each register.

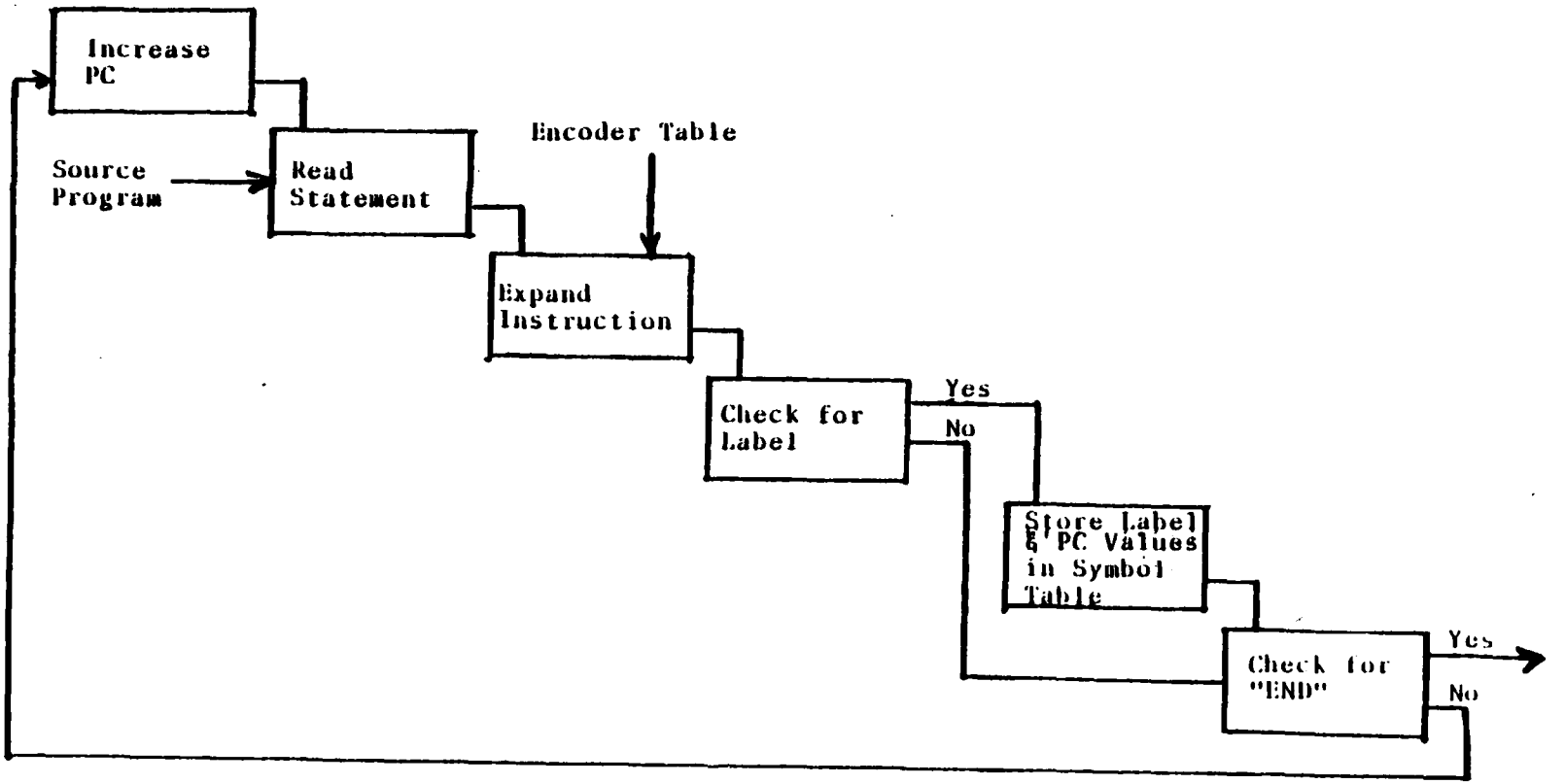


C2A1 Build Identification Section

C2A2 - PROCESS MICROCODE INSTRUCTIONS

Processing the microcode instructions on Pass One of the assembler is necessary in order to give addresses to all labels in the microcode module. The input to this procedure is the microcode source program and the partially completed Symbol Table. The Encoder Table is needed to correctly increment the PC. The output of this procedure is a Symbol Table, ready to be used by Pass Two of the assembler.

Each microcode statement is located in the Encoder Table, and the corresponding delta-value for the PC is found. The delta-value reflects the number of words needed to expand this instruction. A check is made for a label in the statement. If a label exists, the label and current value of PC is stored in the Symbol Table. The instruction is then checked to see if it is an "END" statement. If it is, then Pass One ends. If not, the PC is incremented by the delta-value, and the process is repeated for the next microcode instruction.

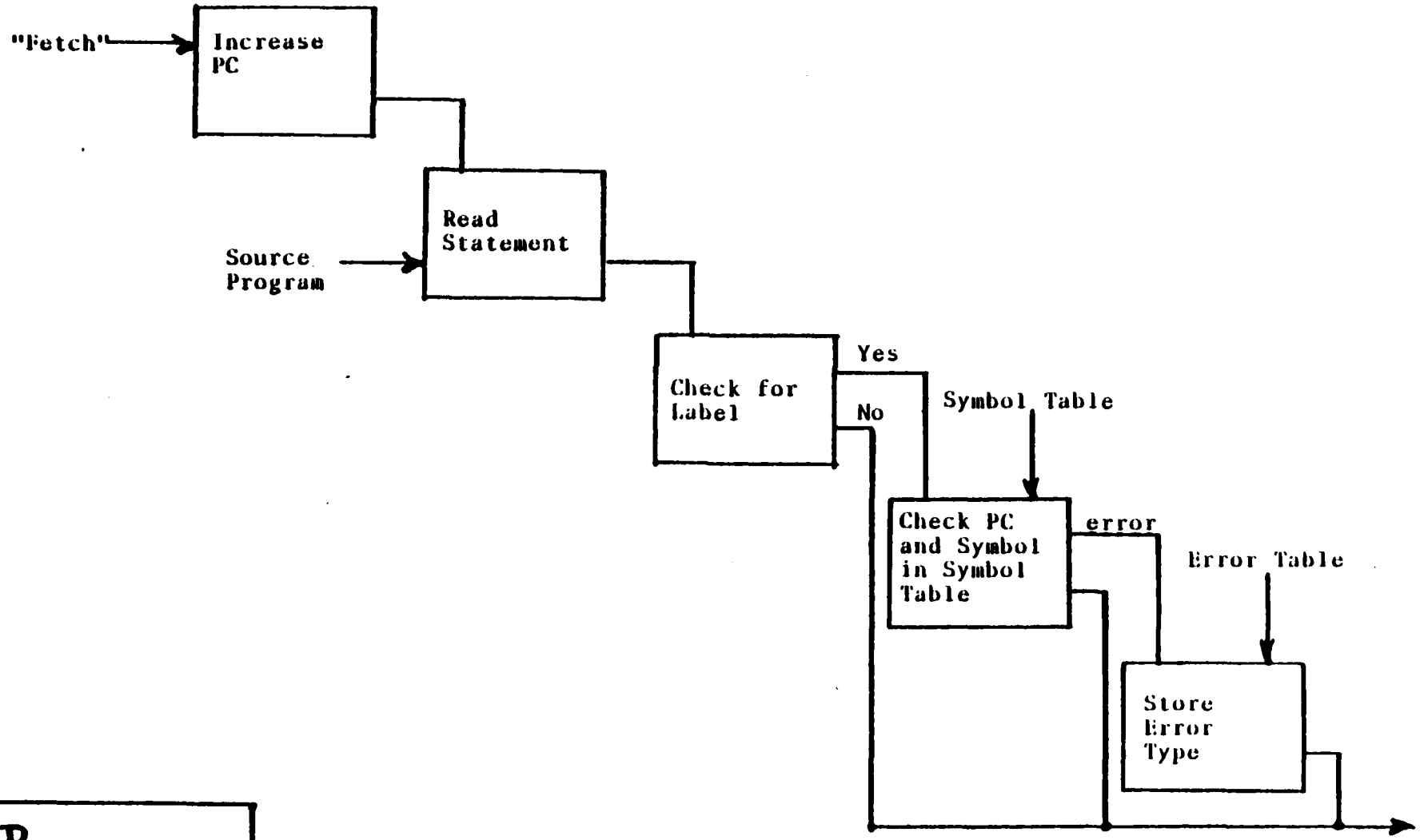


C2A2 Process Microcode Instructions

C1B - PROCESS LABEL

The processing of the Label on Pass Two is done to insure that the current value of the PC is the same as the address of that Label stored in the Symbol Table. If there are any differences, that error is stored in the Error List.

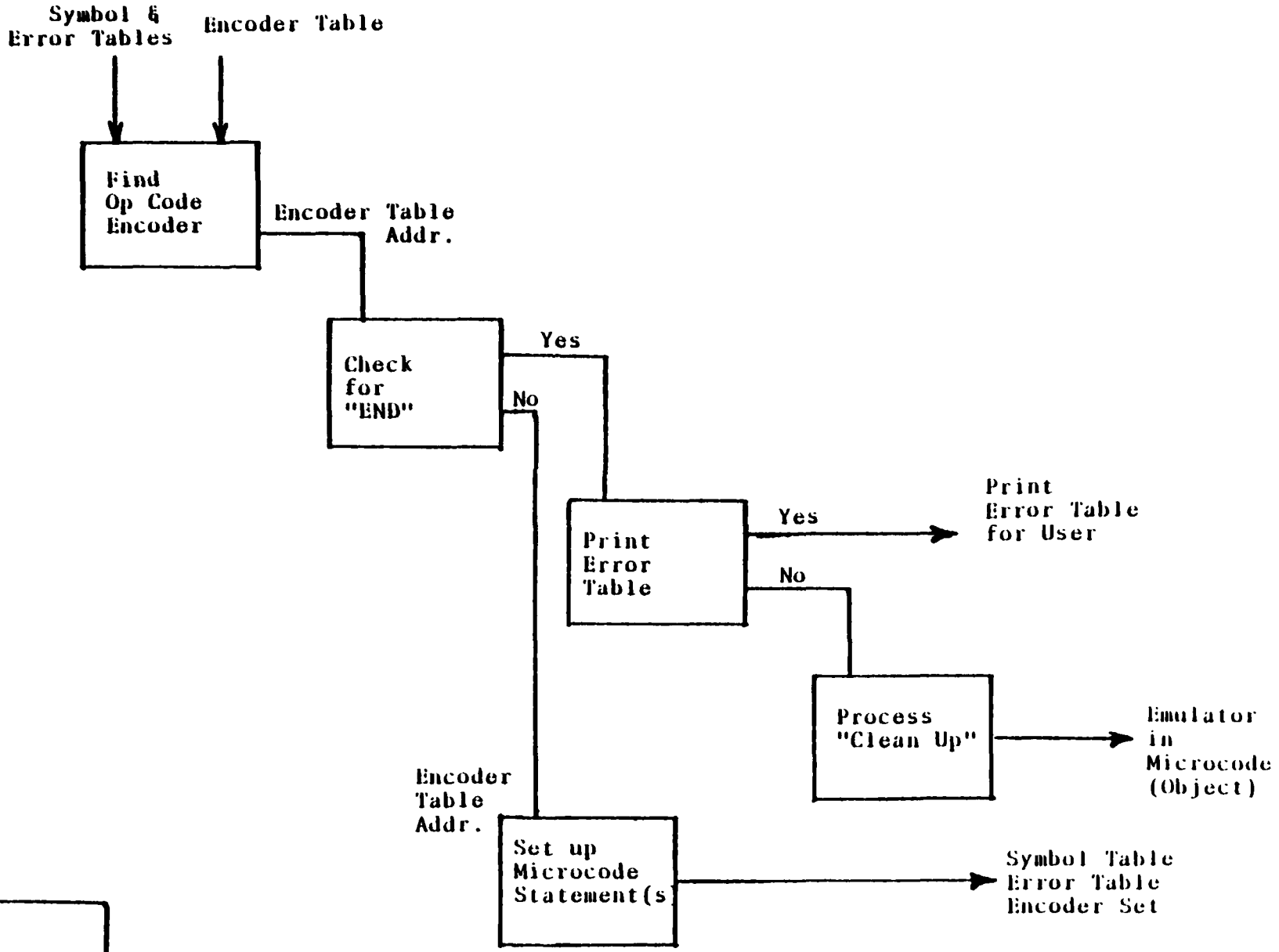
This checking process involves: increasing the PC so that it reflects the correct address of the instruction being processed, reading a statement from the microcode source module, and checking for a Label on that statement. If there is a Label, the Symbol Table is searched for the Label and the current PC is checked against the Symbol Table's value. Any errors are stored in the Error List. If there is no Label in the statement, all checking is by-passed.



C1B Process Label

C1C - PROCESS OPCODE

The Process OpCode procedure takes as input the microcode statment that was read in the Process Label section of the assembler. The tables used are the Symbol Table, the Encoder Table, and the Error Table. The output of this process is twofold. For an individual statement, the binary code which the statement represents is passed to the Process Operand section. However, when the statement being processed is an "END" statement, which signifies the end of assembly, the microcode is "cleaned up" and the microcode object module is output.

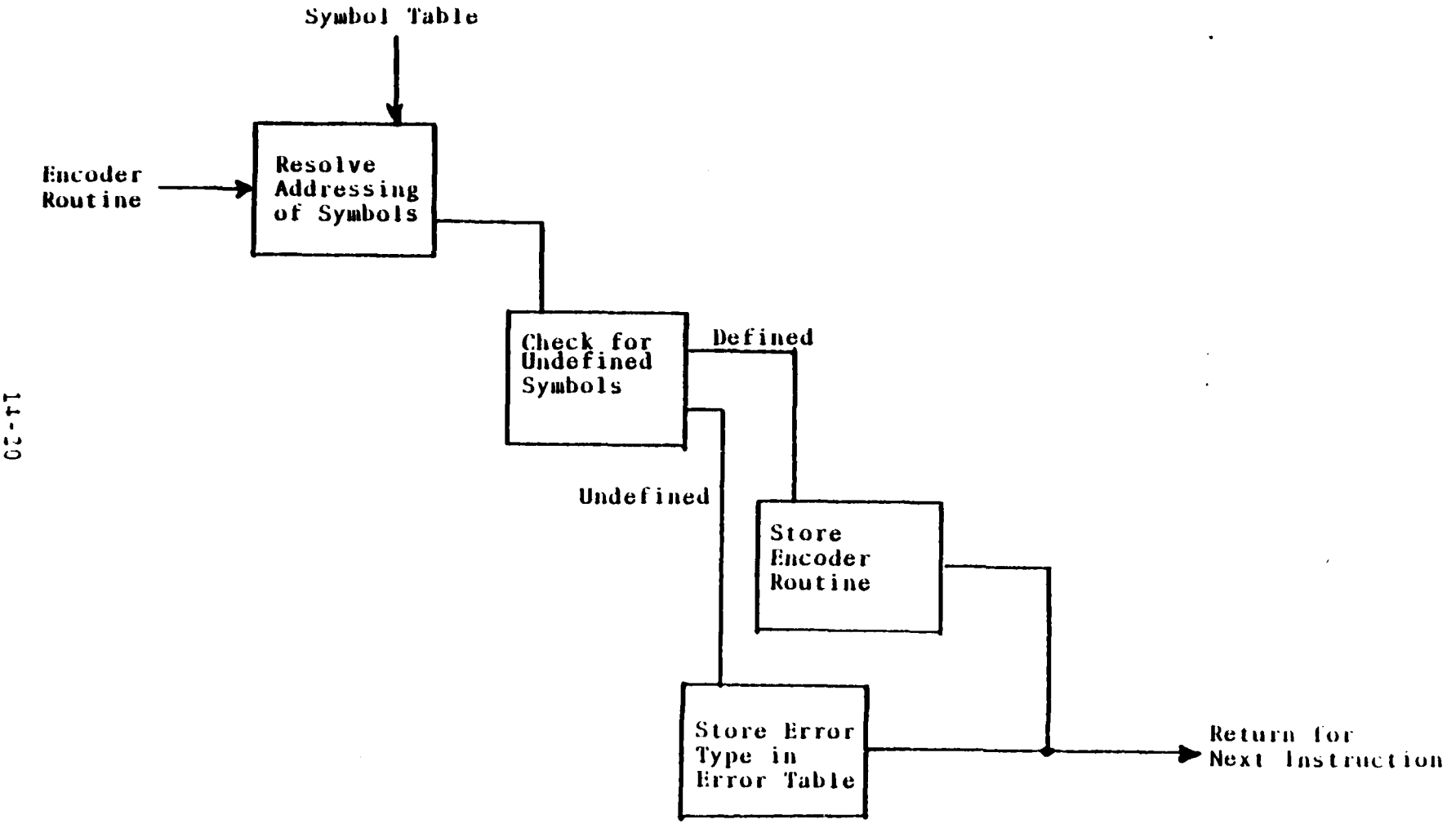


C1C

Process
Op Code

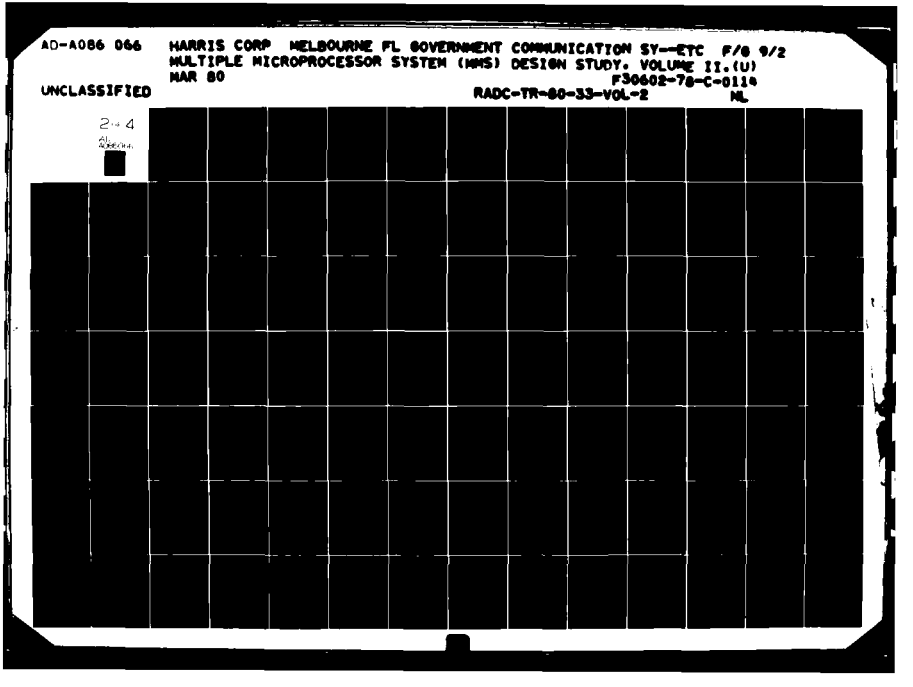
C1D - PROCESS OPERAND

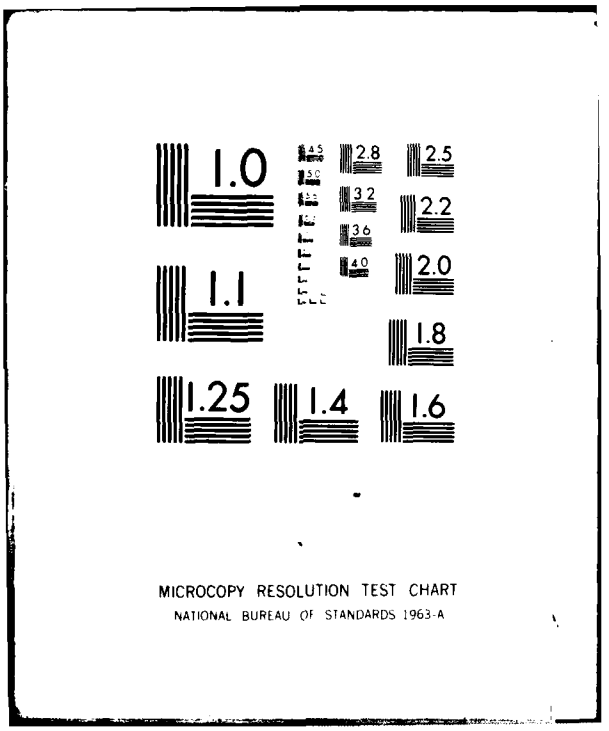
In this Processing section, the microcode statement operands are resolved through use of the Symbol Table. When the value in the Symbol Table is found, the Encoder Routine, which was determined in the Process OpCode section, is updated to contain the addresses of the operands. If the Symbol Table does not contain the operand, the error is noted in the Error Table.



C1D Process Operand

AD-A086 066 HARRIS CORP MELBOURNE FL GOVERNMENT COMMUNICATION SY--ETC F/G 9/2
MULTIPLE MICROPROCESSOR SYSTEM (MMS) DESIGN STUDY. VOLUME II.(U)
MAR 80 F30602-78-C-0114
UNCLASSIFIED RADC-TR-80-33-VOL-2 NL




1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963-A

14.1 Abbreviations Used in Section 14

Addr.	Address
BF	Beginning Flag
Bfr	Buffers
EC	Emulation Code
EF	Ending Flag
FCP	Facility Control Processor
H/W	Hardware
ID	Identification
IEC	Instruction Emulation Code
I/O	Input/Output
MMS	Multiple Microprocessor System
Op Code	Operation Code
PC	Program Counter
PE	Physical Element
PMS	Program Monitoring System
Prt	Ports
RTE	Run-time Executive
SGL	System Generation Loader
Tmr	Timers
TS	Target System

15.0 PE EXECUTIVE (PEXE)

The PE Executive (PEXE) is an operating system which is supplied to the user as part of the MMS software. PEXE is written in microcode and resides in the PE's CS. The purpose of PEXE is to handle the interfacing between the emulated system and the MMS hardware. An interrupt is sent to PEXE whenever the emulated modules need one of the services of PEXE.

Basically, PEXE will initialize itself, the instruction set (IEC), and all the emulated modules. Then, it signals the user at the FCP computer that it is ready for execution. When all PE's are ready, the user can start the execution of the TS. User application programs are executed, with emulated modules interpreting the TS instructions. When these emulated modules need I/O handling, have an error condition, result in a time-out, arrive at the end of the program, etc., they signal PEXE through an interrupt. PEXE will accomplish the desired action, and will return to the emulator when possible.

Figure 15.0-1 shows the charts included to identify the functional design of PEXE.

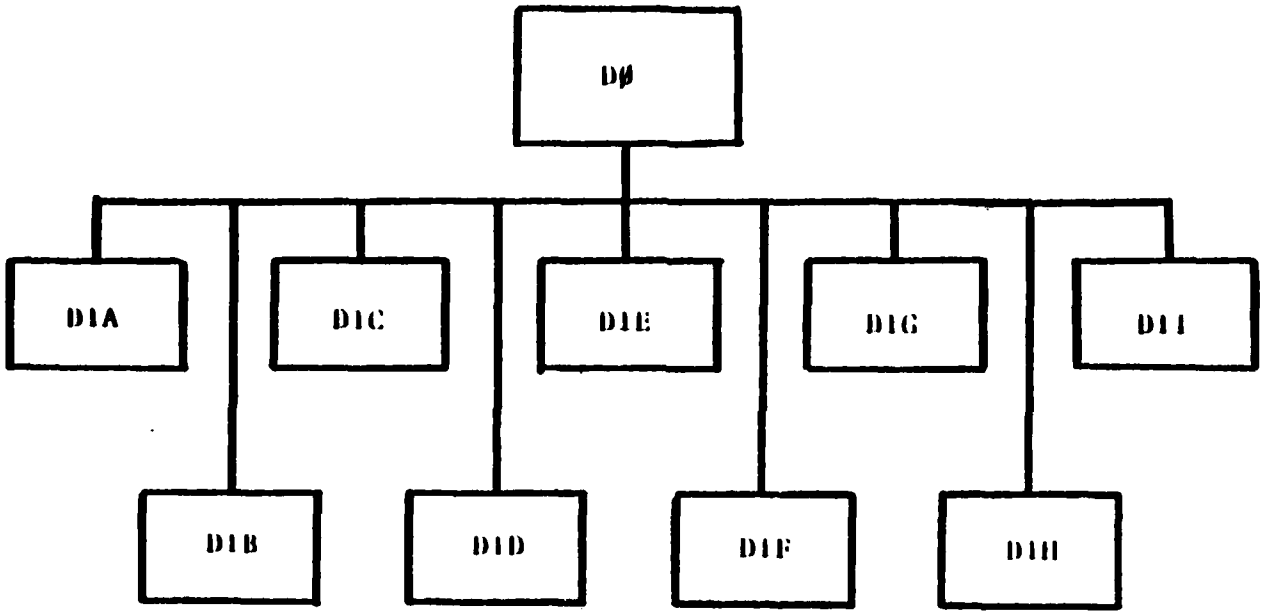
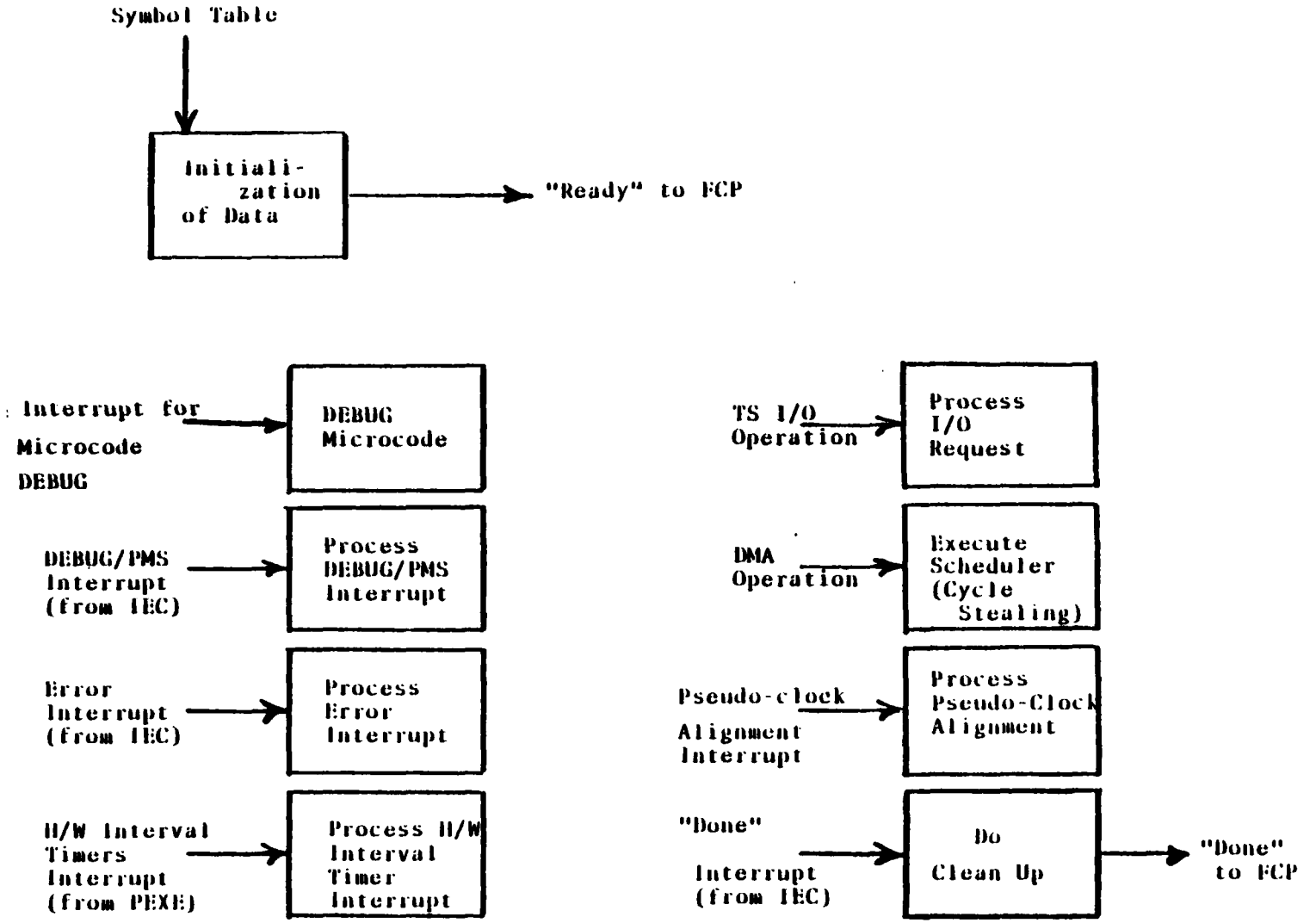


Figure 15.0-1 Top-Down Design Chart

D0 - PEXE

PEXE consists of nine types of operations. One is initialization of data areas for PEXE, the instruction set, and all required emulated modules. The, PEXE has to be able to handle eight types of processes: Microcode DEBUG, PMS/DEBUG, error Interval Timer time-out, I/O, DMA operation,pseudo-clock alignment, and job completion. Each of these eight steps will be discussed individually.


Symbol Table
Initiali-
zation
of Data
"Ready" to FCP
Interrupt for
Microcode
DEBUG
DEBUG
Microcode
DEBUG/PMS
Interrupt
(from IEC)
Process
DEBUG/PMS
Interrupt
Error
Interrupt
(from IEC)
Process
Error
Interrupt
H/W Interval
Timers
Interrupt
(from PEXE)
Process H/W
Interval
Timer
Interrupt
TS I/O
Operation
Process
I/O
Request
DMA
Operation
Execute
Scheduler
(Cycle
Stealing)
Pseudo-clock
Alignment
Interrupt
Process
Pseudo-Clock
Alignment
"Done"
Interrupt
(from IEC)
Do
Clean Up
"Done"
to FCP

DØ

PEXE

D1A - INITIALIZATION OF DATA

The purpose of the initialization step is to prepare the TS for execution. Because the user may wish to restart the TS at any time without having to go through reloading of the total system, this step has to be present.

One approach to letting PEXE know which modules need initialization would be to have all these modules store the addresses to their initialization routine within a special table of PEXE. This would require an additional step per module during SGL.

When all emulation modules are stored within the PE, a mapping table is generated. This table identifies every module, its start address within Control Store for instructions, and its start address within Internal Memory for data. All emulation modules will have to follow a prescribed format, of which Figure 15.0-2 is suggested as a model. Then, the suggested method is that PEXE checks the first data word for every module. If this word contains an address to that module, then this defines the initialization routine for that module. If this first word is zero, then no initialization is required for the module. The second word would then have to be the address to the routine itself.

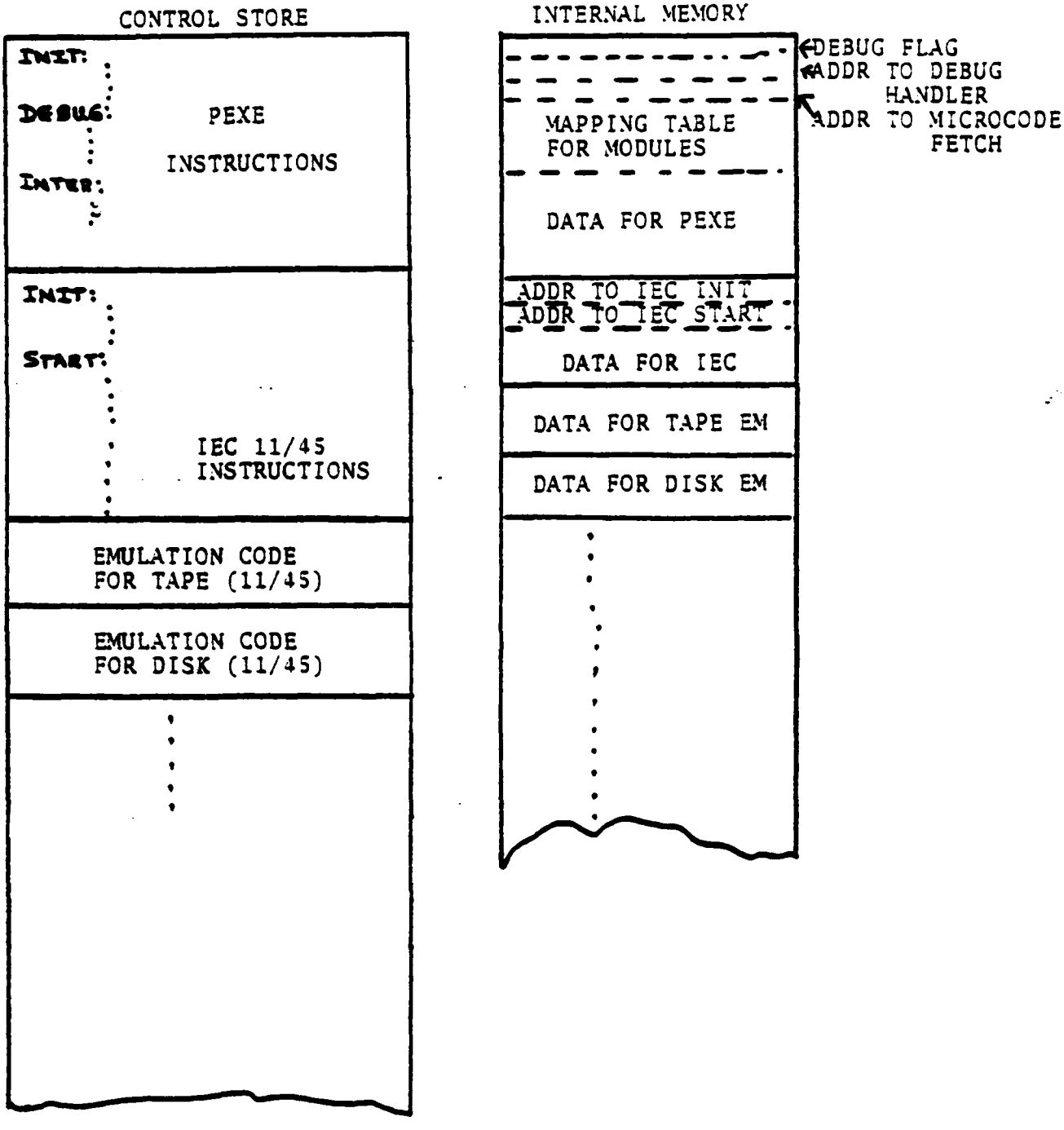
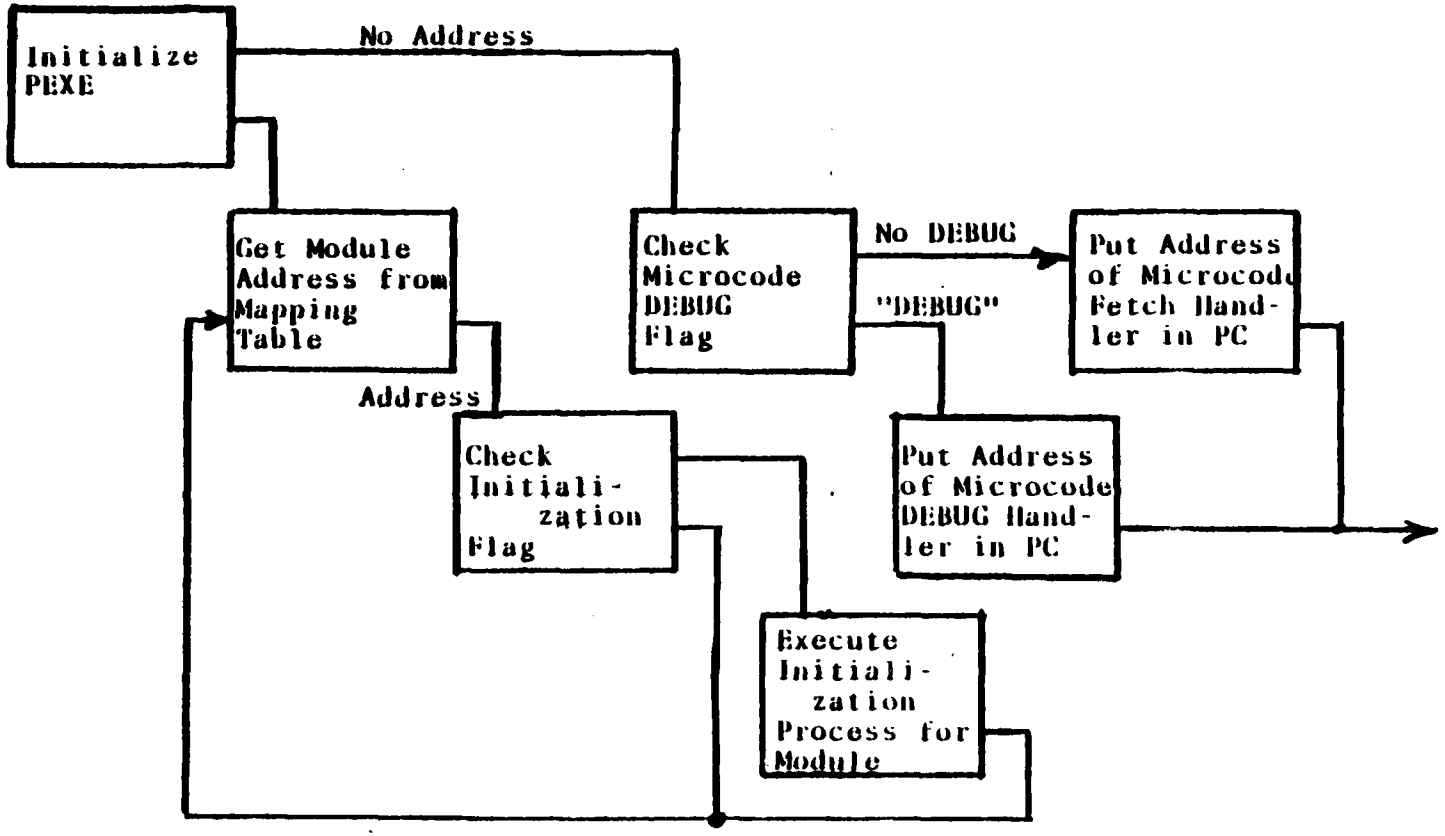


Figure 15.0-2 Suggested Format for Emulation Modules



D1A

Initialization of Data

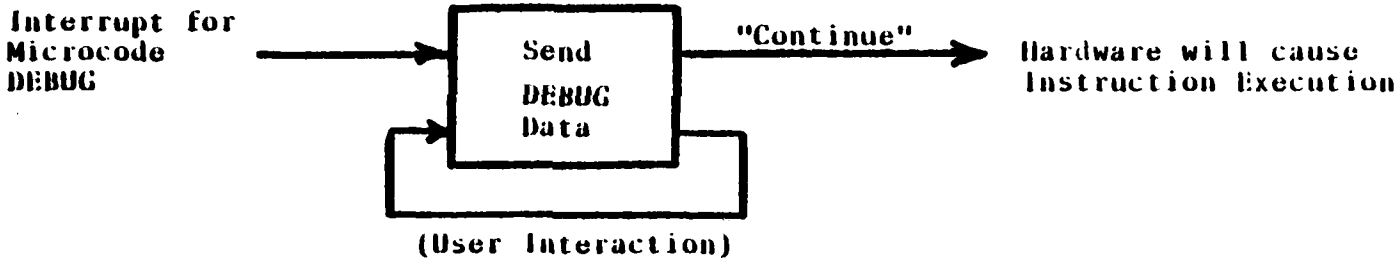
D1B - DEBUG MICROCODE

"DEBUG MICROCODE" is a DEBUG handler for microcode level instructions. This is a handler usable by the microcode programmer and not by the general user.

Hardware is present within the PE to distinguish between Run mode and DEBUG mode. In the Run mode, microinstructions are executed as the Sequencer dictates. When DEBUG of microinstructions is indicated, a specific bit in the microinstruction is set. This action causes the H/W to execute only one microinstruction at the address specified by the Sequencer output. Control is then given to this part of PEXE.

A table of breakpoints is then consulted to see if the user wants to be in single-step mode or if this is a true breakpoint. If none of the conditions are met, then control is returned so that the microinstruction can be executed. If a condition is met, then the DEBUG results are sent to the user. The user can then do a "stop", a "continue", execute n instructions, execute for specified period of pseudo-time, or various dump and load status commands.

Since microcode DEBUG is processed in this manner, it will be very slow. The user should judiciously select DEBUG options at this level because of the great slow-down of execution speed.


Interrupt for
Microcode
DEBUG
Send
DEBUG
Data
"Continue"
Hardware will cause
Instruction Execution
(User Interaction)

D1C - PROCESS DEBUG/PMS INTERRUPT

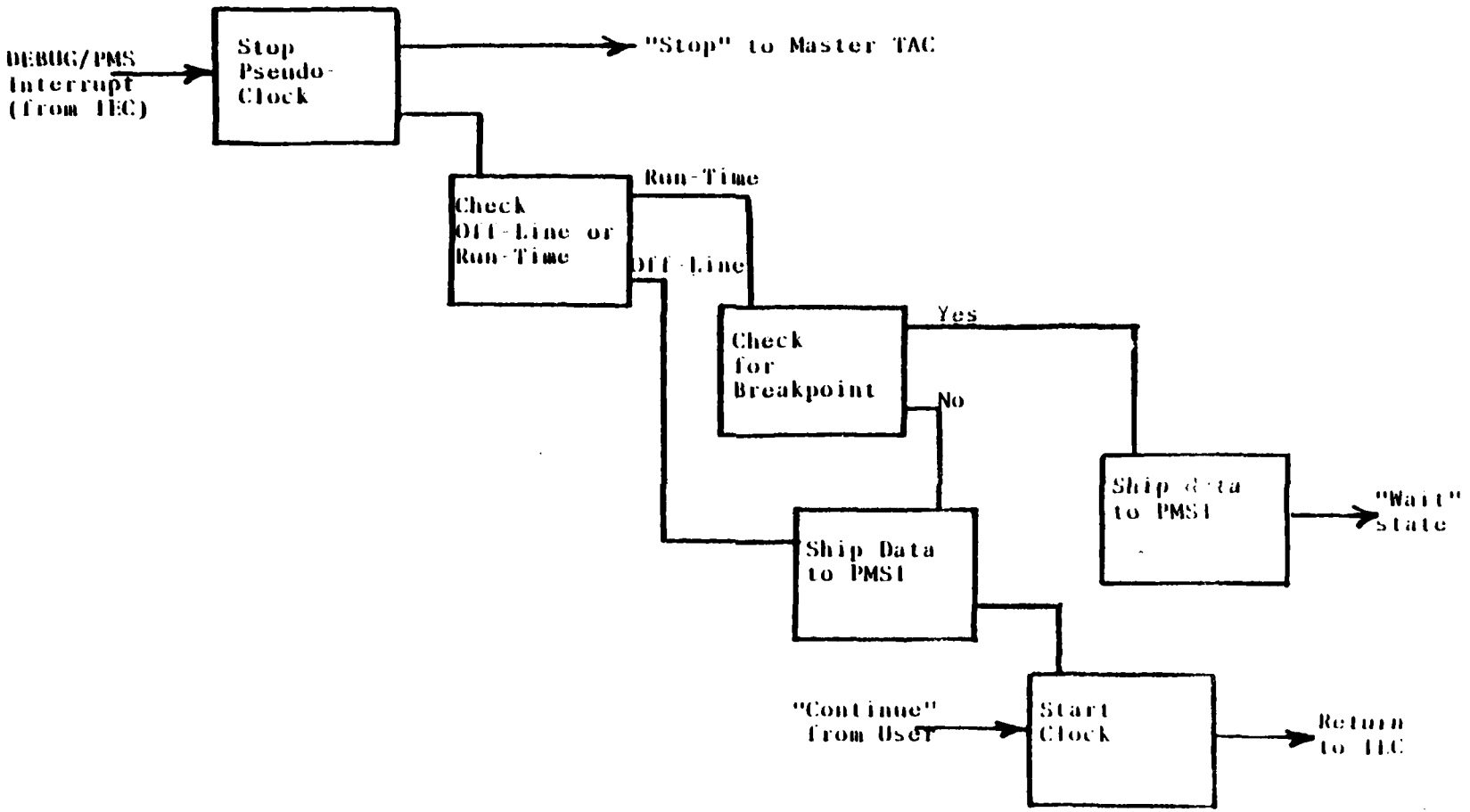
Whenever a macro-level program, I/O process, memory access operation, etc., has ascertained that the DEBUG/PMS flag has been enabled for that operation, that operation will send an interrupt to PEXE. PEXE will then handle the data collection process through PMSI to the desired location in FCP memory.

Upon entering PEXE, because of the interrupt, PEXE will stop the pseudo-clock by sending a stop message to TAC. If the data collection is run-time and the event was not caused by a breakpoint, the data is sent to PMSI. Since PMSI will handle the data and will stop the clock if too much data is sent to it, PEXE will start the clock and will return to IEC. Only IEC can cause an interrupt to PEXE for PMS/DEBUG data collection.

If the event is run-time and is caused by a breakpoint, the data is sent to PMSI for processing, but PEXE has to wait to see what the user may now wish to do.

The user may accomplish several types of actions during a breakpoint, which would cause control to return to other places. If the user wishes to continue at this place in PEXE, the clock is started; control then returns to IEC.

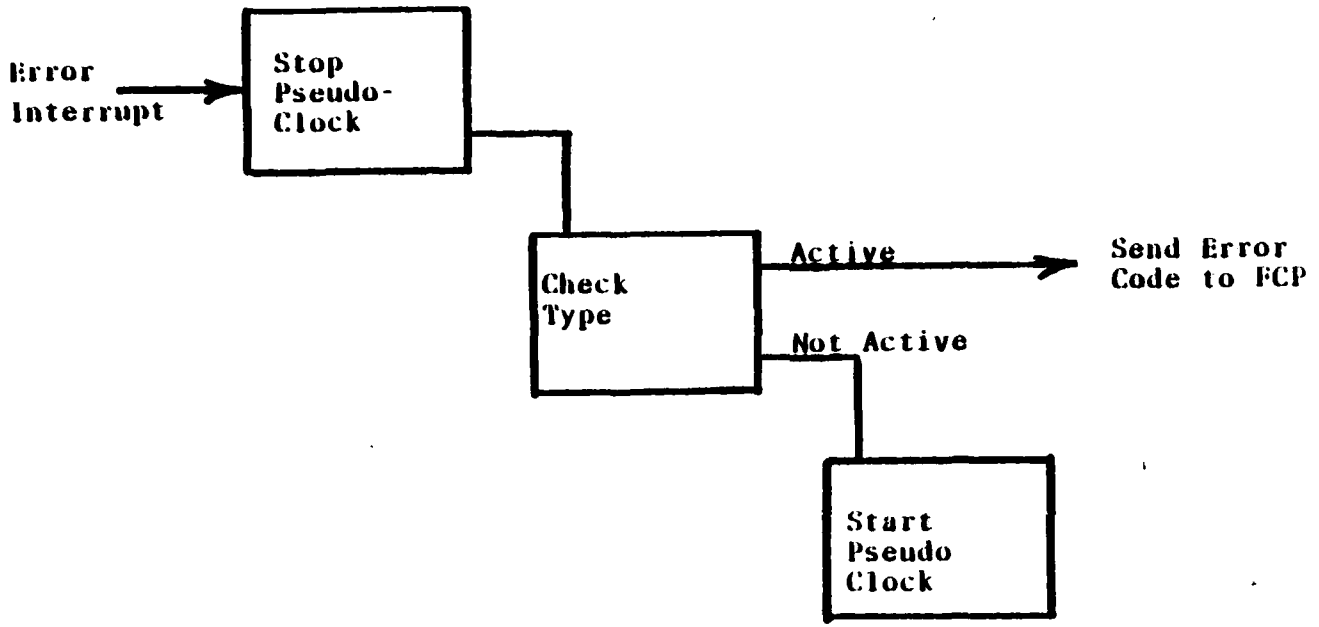
If the user had selected to do off-line data collection, the data needed is sent through PMSI to buffer files within FCP's memory. After PEXE sends the data to PMSI, PEXE starts the clock, and returns control to IEC.



D1C Process DEBUG/PMS Interrupt

D1D - PROCESS ERROR INTERRUPT

Whenever the hardware or software of MMS finds an error, an interrupt to PEXE will handle the processing of this error. The hardware can find several types of errors: parity error, bus timeout, etc. Software can handle several types: access to non-emulated memory, access to memory not assigned to user, access to module not emulated, etc. Many other types of errors can be controlled for both hardware and software. The suggested procedure would be that each type of error be identified by a code. The user would identify those error types to be active for his run. An active error type would stop the execution of the TS. The FCP would identify the error code and type to the user.



D1D Process Error Interrupt

D1E - PROCESS H/W INTERVAL TIMER INTERRUPT

The emulation modules will have one of the 16 Hardware Interval Timers assigned to it. When this module is executed, the real execution time is stored in its Interval Timer. When this Timer decrements to zero, an interrupt is sent to PEXE.

PEXE will stop the Pseudo-clock while it processes this interrupt. First, PEXE will determine which Timer caused the interrupt. If Timer Ø caused the interrupt, then all Software Interval Timers are decremented. These exist as a stack of Software Timers which are decremented only when Hardware Timer reaches zero; thus, they are set as multiples of the value store in Timer Ø. There may not be any Software Interval Timers, because these are assigned only when there are not enough Hardware Timers. So, after all Software Interval Timers have been decremented, all are checked to see if any have the value of zero. If so, then the address of correct handler is found and control transfers there. When control returns from the Handler, that Software Interval Timer is reset and the pseudo-clock is restarted.

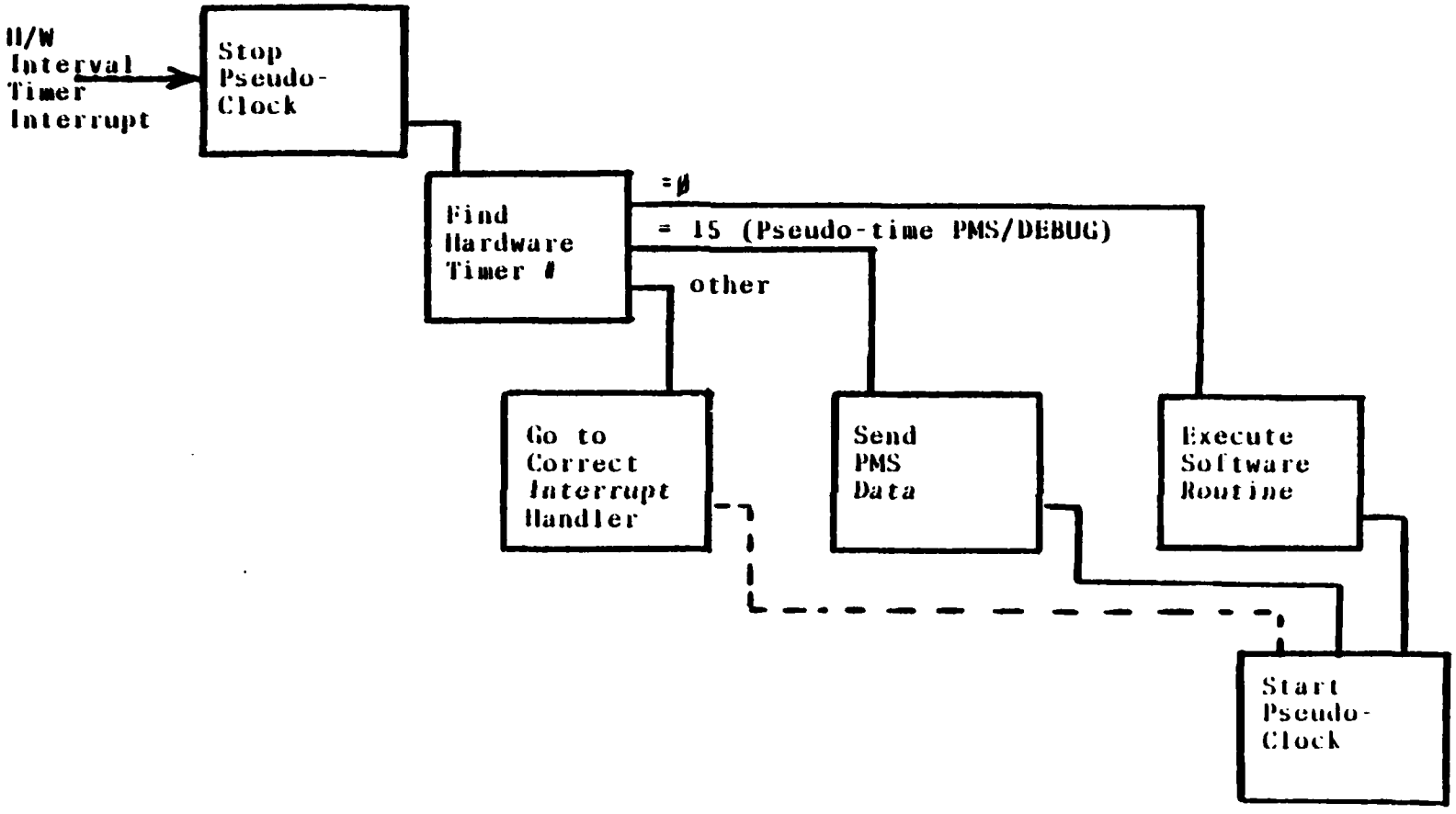
If Hardware Interval Timer 15 caused the interrupt, then PMS data collection based upon pseudo-time is to be done. Thus, PMS data is sent to PMSI and the pseudo-clock is restarted.

For all other Hardware Interval Timers, control is transferred to the user-assigned Interrupt Handler. Then the pseudo-clock is restarted.

In all cases, a default Interrupt Handler is part of PEXE. This Handler would allow the instruction process being emulated to continue until finished, without further values from

the pseudo-clock. Then, when the emulation process is done, control returns to the step in this phase where the pseudo-clock is reset.

Also, the Hardware Timers are automatically reset. The Software Timers have to be reset by software.

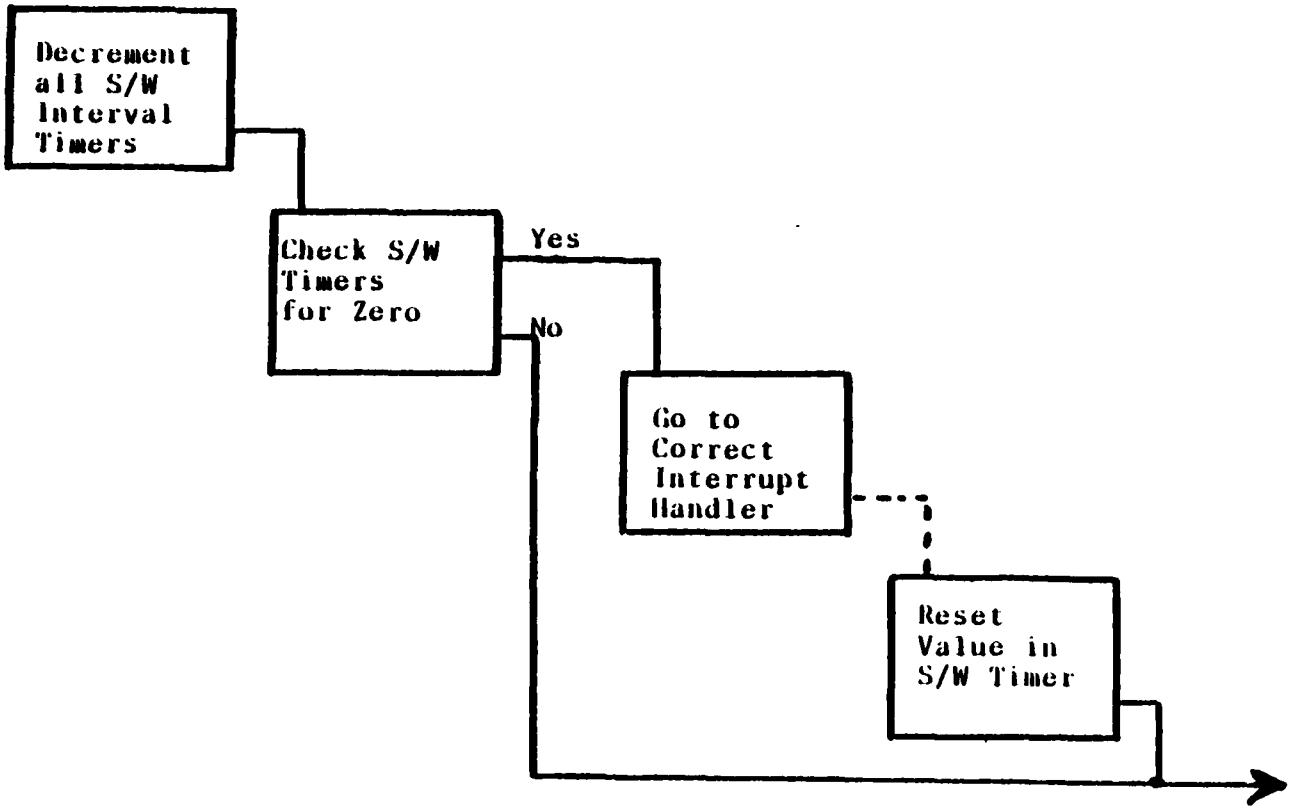


D1E Process H/W Interval Timer Interrupt

D2E5 - EXECUTE SOFTWARE ROUTINE

If some Software Interval Timers have been assigned, they are all decremented whenever Hardware Interval Timer Ø reaches zero and causes an interrupt to PEXE. Then, all the Software Interval Timers are checked to see if any of these have been decremented to zero. If not, then control returns to the step in D1E ("Process H/W Interval Timer Interrupt") where the Pseudo-clock is restarted.

If one of the Software Interval Timers now contain zero, then the Handler for that process is executed. The logic of the Handler is user-designed; if not added by the user, PEXE will have a default address to go directly to the next step, "Reset Value in S/W Timer". Once the Handler is finished, the value is reset into the Software Timer.



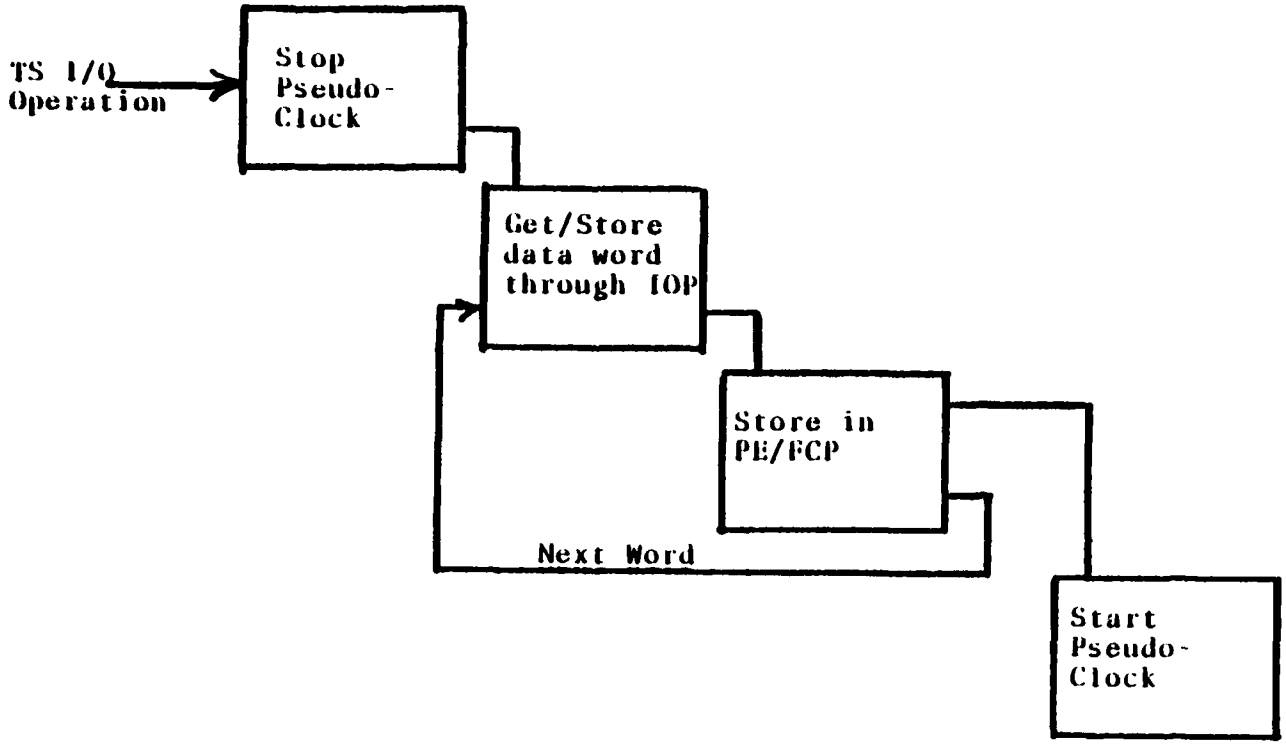
D2 E5

Execute Software Routine

D1F - PROCESS I/O REQUEST

The hardware emulators for I/O processes can send this type of interrupt when the user wishes to transfer a large block of data from the FCP to someplace within the PE for TS usage, or from the TS in the PE to FCP memory.

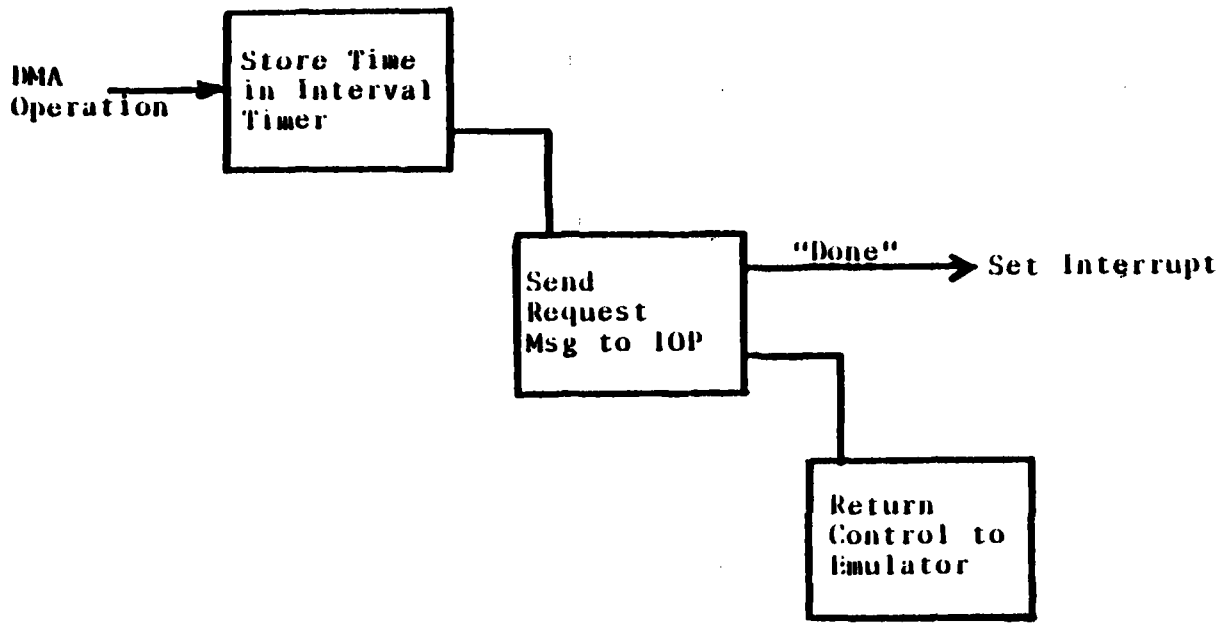
First, the pseudo-clock is stopped. Then the transfer of the data is done one word at a time over the MMS. The data can be from memory of the FCP to the TS to emulate input. The data can be from the TS to a memory buffer in the FCP as output. When the total transfer is completed, the pseudo-clock is started.



D1F Process I/O Request

D1G - EXECUTE SCHEDULER

The Scheduler is a user-selectable routine. The purpose for having the Scheduler is so that the user can do I/O by interleaving the accesses with other instruction executions. This interleaving is called cycle stealing. The reading or storing of data is one word at a time. When each word process is completed, an interrupt is set so that the TS CPU is aware that this process is ready for restart. Meanwhile, control is returned to the requesting emulator so that it can continue processing.

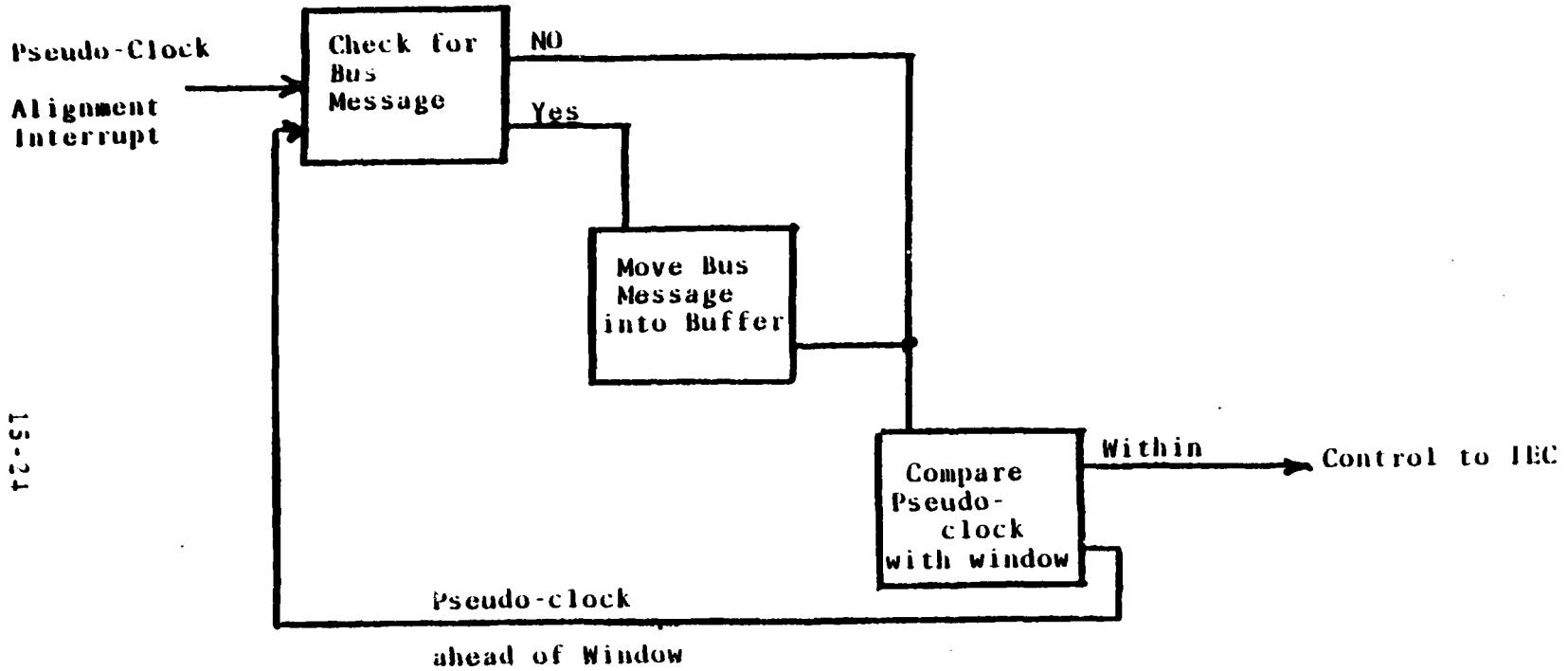


D1G Execute Scheduler

D1H - PROCESS PSEUDO-CLOCK ALIGNMENT

A Pseudo-clock Alignment Interrupt is generated whenever the PE's Pseudo-clock is ahead of the Window. The PE has to remain in an idle state and cannot execute any more macro instructions until the other PE's of the TS can catch up.

If any Bus Messages have been accepted by the BIU of the PE, this Handler will save the message on a stack (or Buffer). Then the BIU is free to send or receive any other Bus Messages. Also, this Handler will check to see if the Window is now in alignment with the PE's Pseudo-Time clock. If yes, then control is transferred to the IEC so that it can start execution again of a macro instruction. If no, then the Handler loops back to again check for Bus Messages.

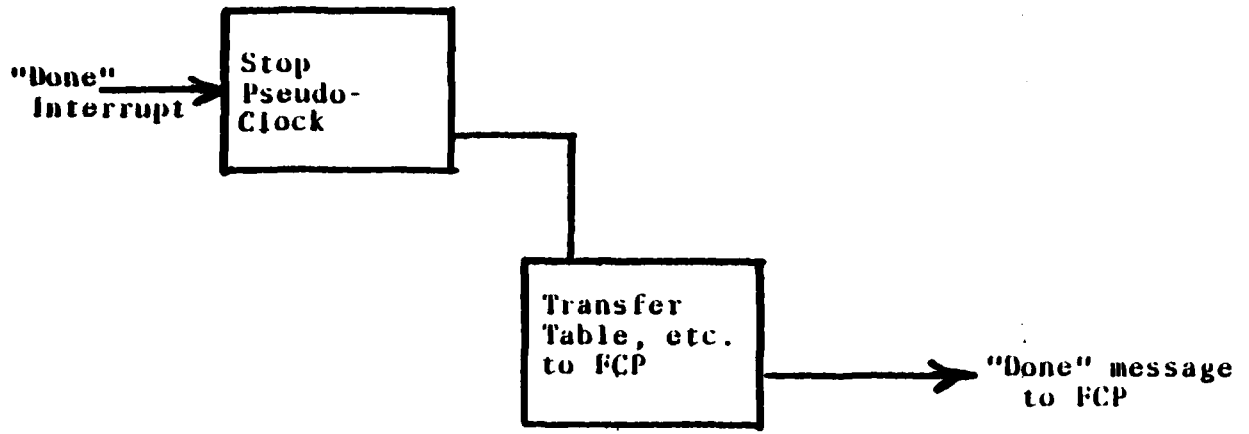


15-21

D1H Process Pseudo-clock Alignment

211 - DO CLEAN UP

The pseudo-clock is stopped. Whatever is needed by FCP to complete data collection is transferred; i.e., tables, ending pseudo-time, etc. Then the "Done" flag is sent to the FCP, where final MMS housekeeping is done.



D1I

DO CLEAN UP

15.1 Abbreviations Used in Section 15

ADDR	Address
BIU	Bus Interface Unit
CPU	Central Processing Unit; CPU of emulated Computer
CS	Control Store
DMA	Direct Memory Access
EM	Emulation Module
FCP	Facility Control Processor
H/W	Hardware
IEC	Instruction Emulation Code
I/O	Input/Output
IOP	Input/Output Processor
MMS	Multiple Microprocessor System
PC	Program Counter
PE	Processing Element
PEXE	PE Executive
PMS	Performance Monitoring System
PMSI	PMS (Performance Monitoring System) Interface
SGL	System Generation Loader
TAC	Time Alignment Controller
TS	Target System

16.0 SYSTEM GENERATION LOADER

The System Generation Loader (SGL) is the first software module utilized by the user to configure the MMS system. The purpose of SGL is to aid the user in selecting all the hardware packages which emulate the I/O portions and which emulate the instruction set of all computer systems which comprise a Target system (TS).

The user utilizes the software package SGL as it resides on the Facility Control Processor (FCP, which is a mini-computer). Through SGL, the user scans the files of emulation modules which help to define the computer systems of his Target System. Once the User has made the selection of those modules which emulate the peripherals of his computer system, SGL will download the modules to the PE's of MMS. The linkages are, as much as possible, automatically developed. If necessary, the user is questioned during SGL to help finish the necessary linkages. Then, these linkage tables are saved to help link all emulated Computer Systems into the desired Target System.

SGL can be accessed by the user in an interactive mode, or by an interactive quick-load mode, or through a control file. These methods are discussed thoroughly in the section relative to "User Scenario", Section 21. Thus, SGL will be a large, powerful, software module. Its name implies its purpose: to generate the TS emulation and to then load the TS so that it is ready for execution.

Figure 16.0 sets forth the diagrams included in this section to accomplish the software design.

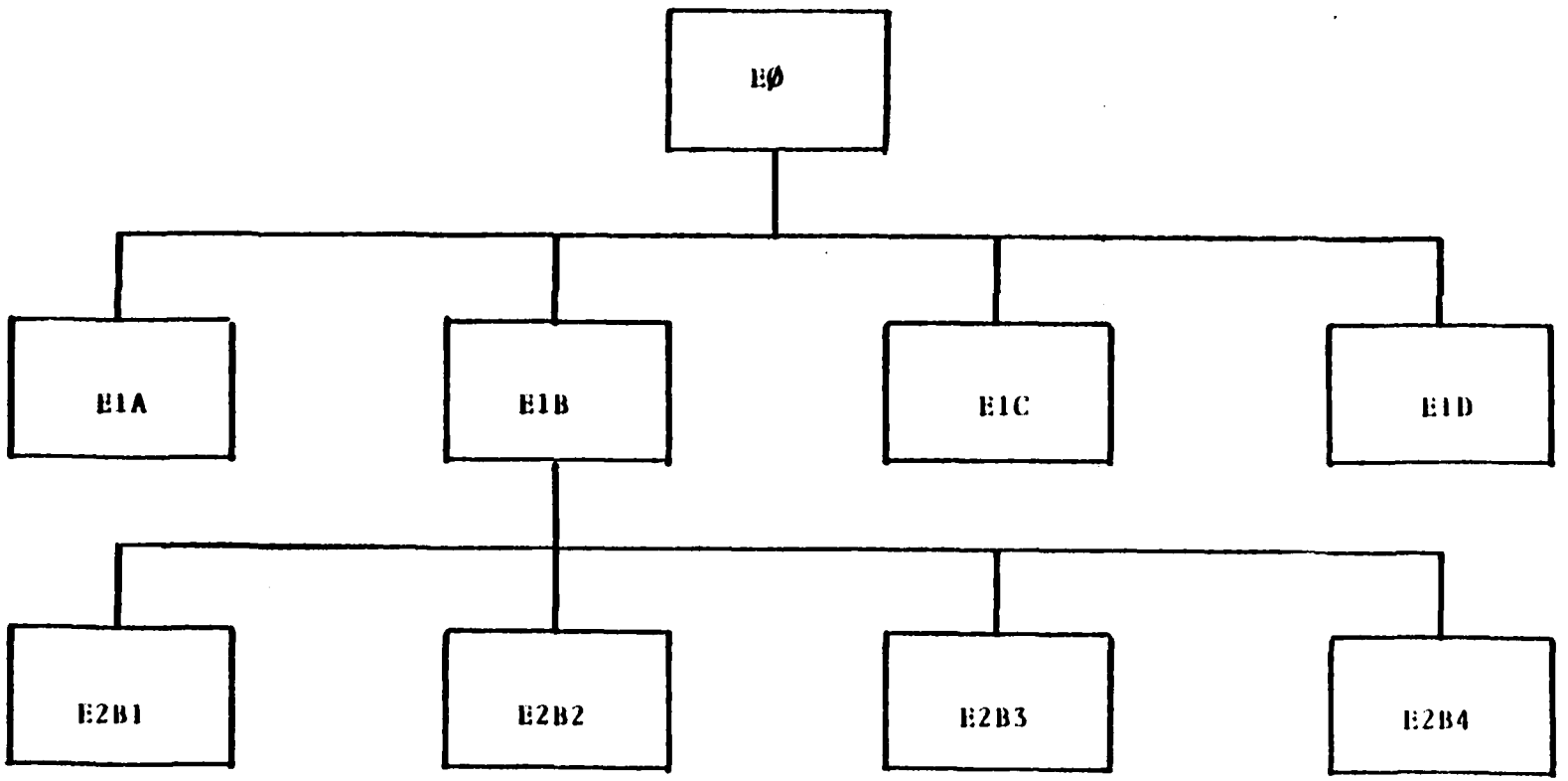


Figure 16.0 Top-Down Design Chart

EØ - MMS SYSTEM GENERATION LOADER (SGL)

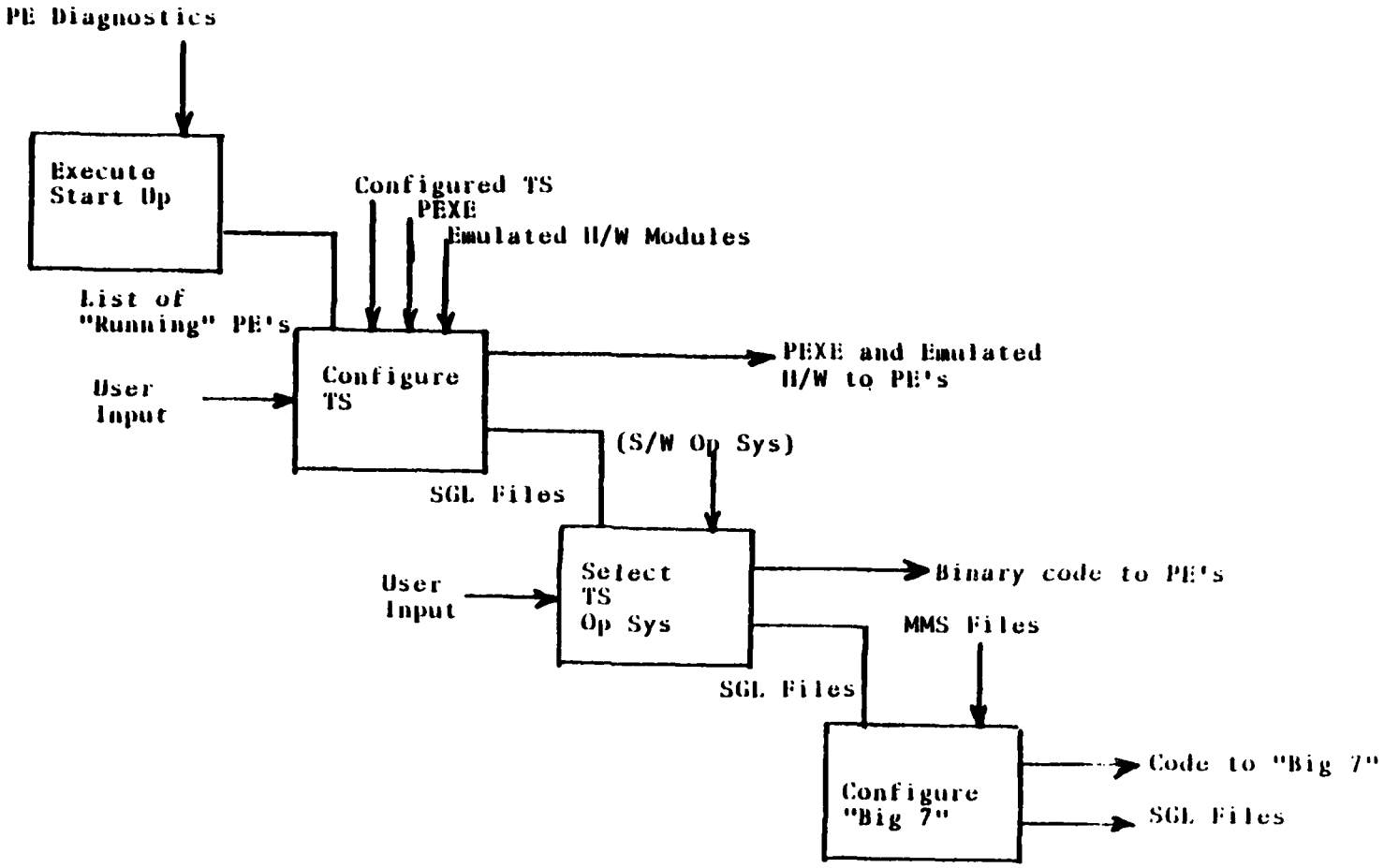
SGL is the first program that the user calls so that he can use MMS. SGL will use the hardware diagnostic package to verify the total number of PE's (Physical Elements or microprocessors) that are operational. This allows users, who need many PE's for their emulations, to verify that enough PE's are operational.

Then SGL allows the user to select:

- A Target System (TS) that has been completely configured, as a network of several computers.
- One or more computer systems that have been completely configured, e.g., an IBM 370/155 and a PDP-11/45.
- Which components he desires from those available in order to configure his own computer system.
- Which interconnects he desires between any emulated computer systems, in order to design his own network system.

A vendor supplied executive and the emulated modules are downloaded to the PE's by SGL. All linkage is accomplished by SGL. When the TS involves a shared resource or needs access to an external device which is emulated on the FCP (Facility Control Processor) computer, SGL also does the necessary linkage.

The output from SGL consists of the TS configuration being added to the Library File for future utilization, the MMS hardware being loaded with the emulators of the TS, and a set of SGL Tables to be available for the Run-Time Executive. These SGL Tables will be mapping tables necessary to link in the TS software and to execute the user programs with, or without, DEBUG/PMS.


PE Diagnostics
Execute Start Up
Configured TS
PEXE
Emulated H/W Modules
List of "Running" PE's
Configure TS
User Input
PEXE and Emulated H/W to PE's
(S/W Op Sys)
SGL Files
Select TS Op Sys
User Input
Binary code to PE's
MMS Files
SGL Files
Configure "Big 7"
Code to "Big 7"
SGL Files

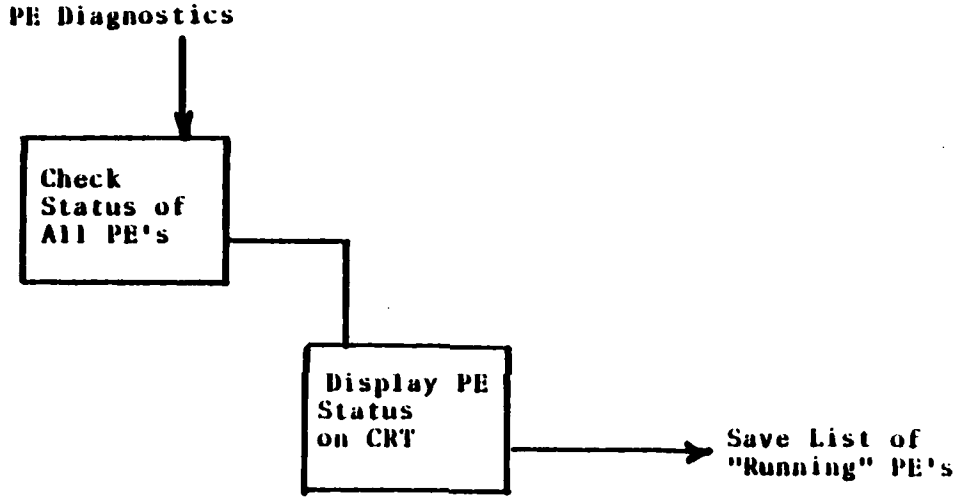
EØ

MMS SGL

E1A - EXECUTE STARTUP

Since the hardware diagnostics are programs to be written as subroutines and since these can be then utilized to functionally check out every PE and all other hardware of MMS, the first step of "Execute Startup" is to verify that all components of MMS are operational. This step will probably take from 30 seconds to 2 minutes. This allows the user to know what is operational before trying to configure a Target System (TS). This data will also need to be saved for further utilization of the PE's.

Of course, this information is to be immediately displayed on the CRT for the user.



E1A Execute Start Up

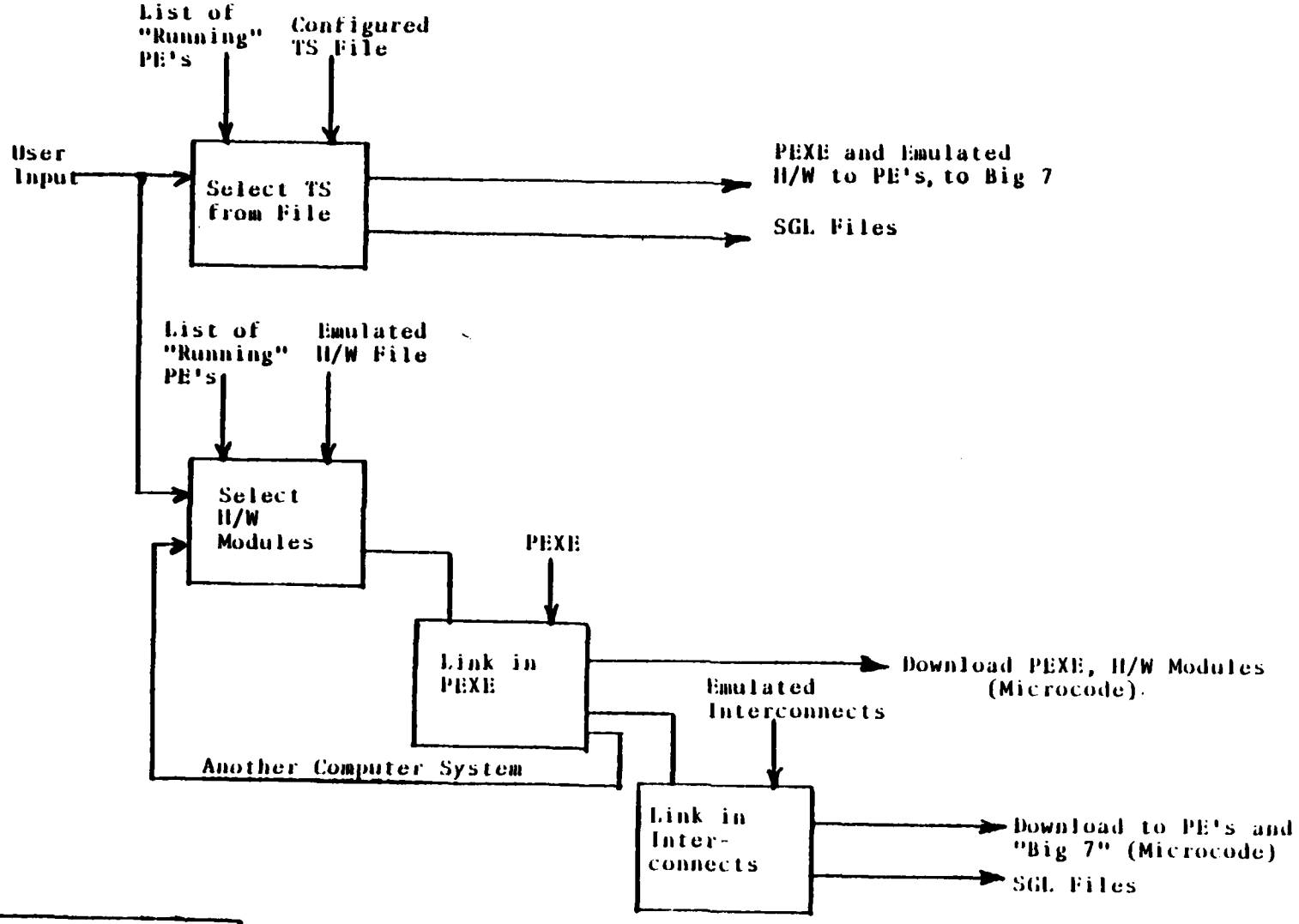
E1B - CONFIGURE TS

Within this step, the user can select and download an existing TS that has been configured and has been saved within the "MMS TS File". A Target System (TS) will consist either of one computer system or of several computer systems linked into a network. Therefore, this "MMS TS File" will include, for each computer system desired, the microcode modules for PEXE, the TS instruction set, all emulated hardware modules, and the necessary emulation of all interconnect interfaces. The interconnect, such as a bus, will have arbitration modules in the Shared Resource Controller (SRC), and the I/O devices emulated on the FCP Computer will be linked through the I/O Interface (IOP). The necessary linkage tables, which remain in FCP memory and allow the Run-time Executive (RTE) to function, are also saved as part of this "MMS TS File".

If the user chooses to configure a new TS, he then "grocery shops" through the library of emulated hardware modules and proceeds to build a TS of one or more computer systems. He can select which PE or set of PE's to use per computer system, and he then identifies the ports of the Computer Systems to tie in to the bus, to any other interconnect, to a shared resource, or to another I/O device emulated on the FCP. As the user designs his TS, these emulated hardware modules are linked with the PE Executive (PEXE) and are downloaded to the PE. If the user designs a TS of several computer systems, he also has to select and connect in the required interconnects.

It is strongly recommended that all emulated packages

written by a microprogrammer also have diagnostic programs for verification of correct linkages by the user programmer. It is recommended that all new TS's be tested for correctness by also running pertinent TS diagnostic programs.



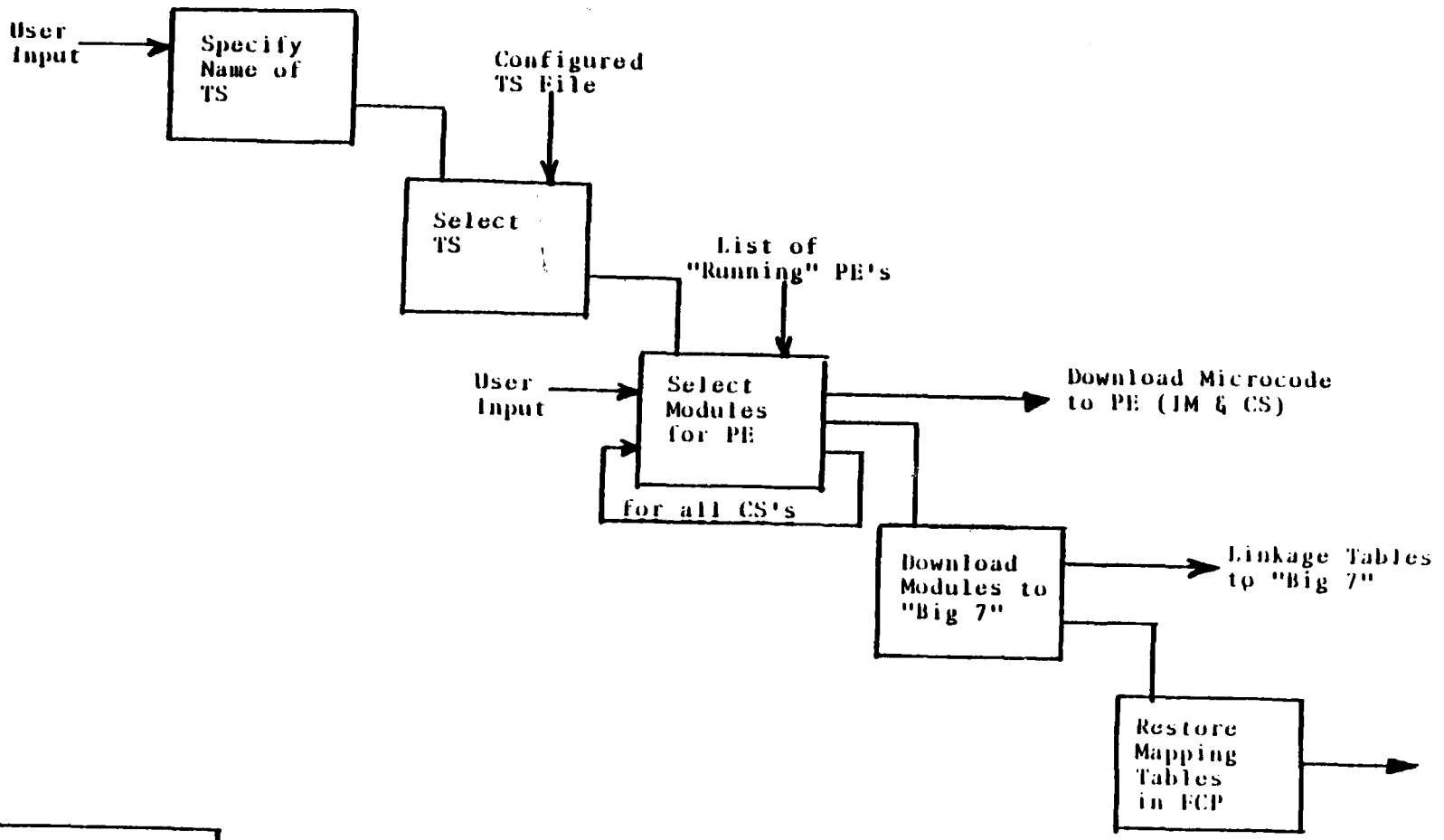
E1B Configure TS

E2B1 - SELECT TS FROM FILE

In many cases, a TS will already be available for the user in the Configured TS File. This will be true if the user has previously configured the TS and has saved the TS configuration. A TS will consist of one or more emulated computer systems linked together in a network. The complexity of a TS depends upon the details of the emulations of the hardware components.

Once the user has selected which TS he desires, the total system is downloaded to the PE's. The user should be able to specify which PE's he wishes to use, or he can let these be assigned automatically. It should be possible to have this TS assign the PE's, but it is possible that these PE's may not be operational. Or, if one PE of a series may not be operational, this might have an effect upon the efficiency of the total TS; i.e., all PE's required should reside on the same segment within MMS to minimize actual bus accesses. It is suggested that, when the user specifies the TS Name, the user be able to call it by number within a list, as from a menu; otherwise, the use of a space at the wrong spot within the name will cause the user name to not match the name within the directory.

The downloading of code to MMS will consist of microcode to Control Store within the PE, to SRC, to PMSI, and to IOP. Linkage tables will go to memory within the FCP Memory. Other linkage tables used by PEXE and all data used by program modules in Control Store will be downloaded to Internal Memory.

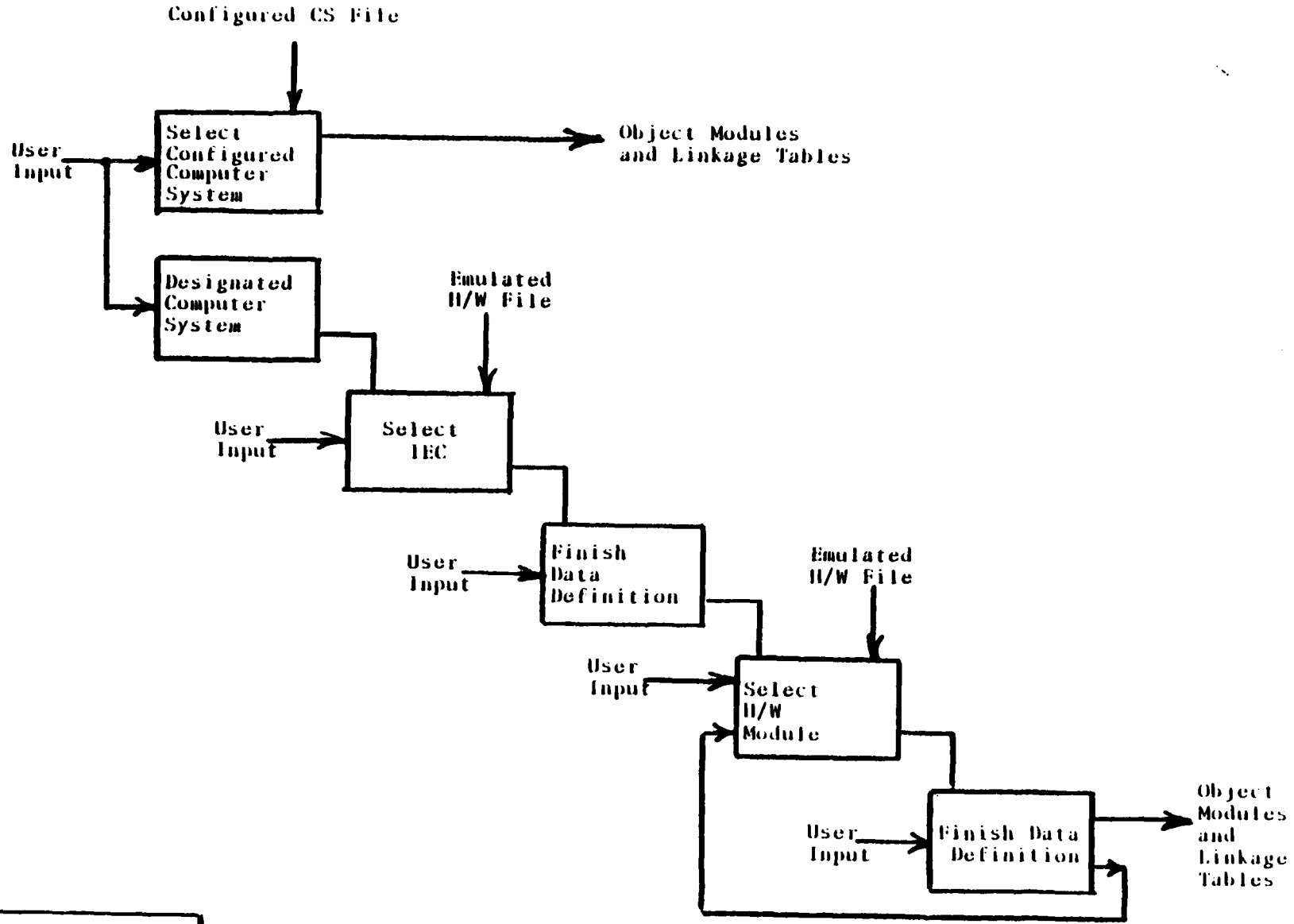


E2B1 Select TS from File

E2B2 - SELECT HARDWARE MODULES

This step, along with the next step, is repeated so that the user may add the emulation of computer hardware systems to his TS design.

The user specifies which computer system he desires to emulate. Then, the library of all emulated packages available for that computer system is listed. The user selects first the IEC, or the Instruction Emulation Code, module. If necessary, the user responds to questions which help to further refine the emulation to fit his requirements. SGL will update the user's preferences into the data area of IEC. Then, the user selects a module which emulates some hardware which the user wants as part of this computer system design. Again, the user may be asked his preference relative to some variables of the emulation, and these inputs will update the data area of this module. The selection of hardware modules continues until the user is finished.



E2B2 Select H/W Module

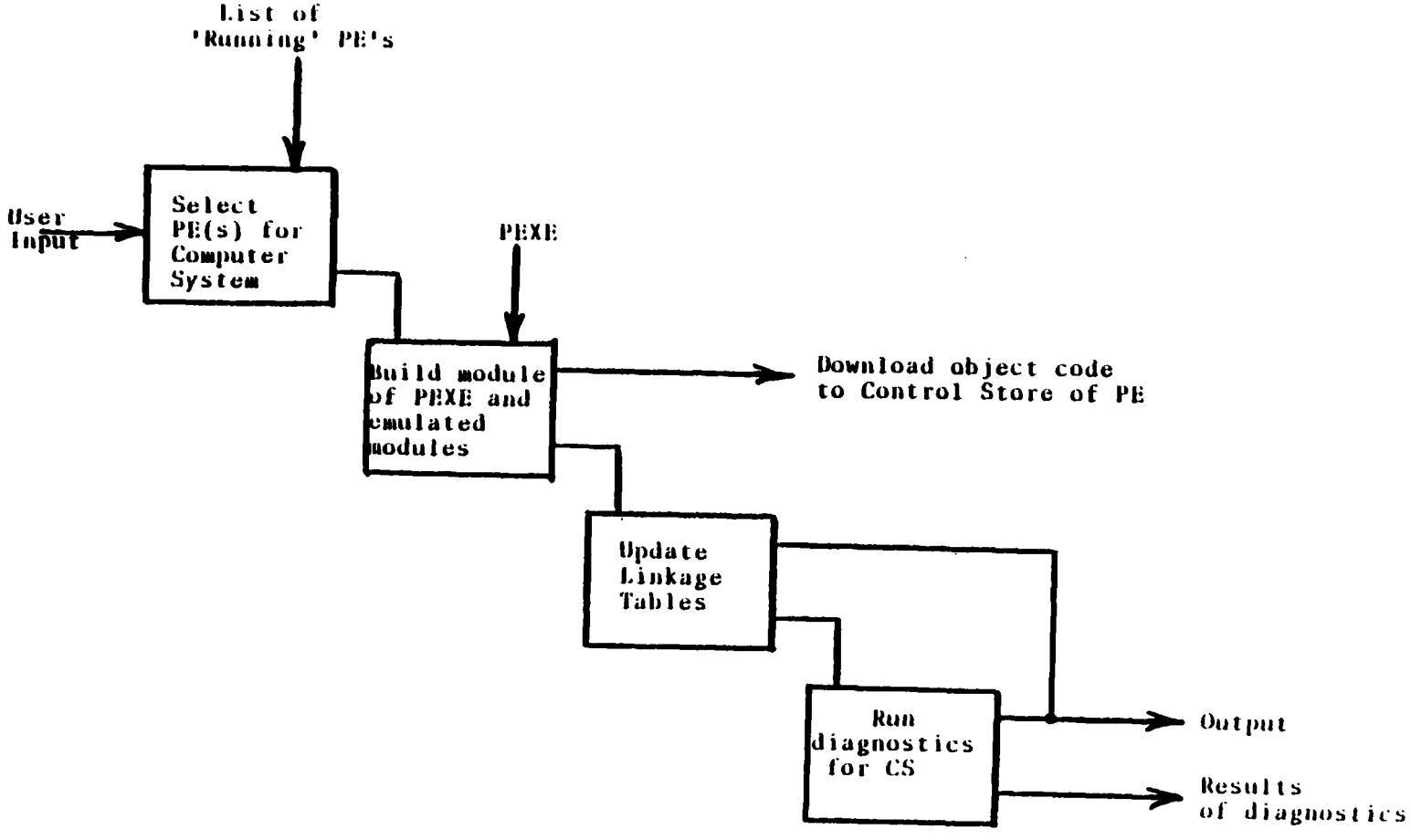
E2B3 - LINK IN PEXE

At this point, the user has selected the IEC and all the emulated hardware modules to specify the characteristics of a computer system. The user can specify which PE's he wishes to utilize, or he can let SGL do this for him. The list of running, or operational, PE's is available for this selection.

The object code is prepared to be downloaded to the PE's Control Store memory. The data areas required by these modules is prepared for downloading to the PE's Internal Memory. The mapping tables for the storage of all of these modules is prepared, in order to enable all future linkages and references to these modules.

The CS and IM are loaded with the object code and the data. The linkage tables for PEXE will be downloaded later because they are of fixed length and are incomplete at this point.

Diagnostic Programs should be processed to check the correctness of emulation and the linkage between modules of a computer system. Some linkage tables will need to be in IM if diagnostics are processed. Diagnostic programs are unnecessary for an existing configured CS, if that CS has already been verified.



E2B3

Link
in PEXE

E2B4 - LINK IN INTERCONNECTS

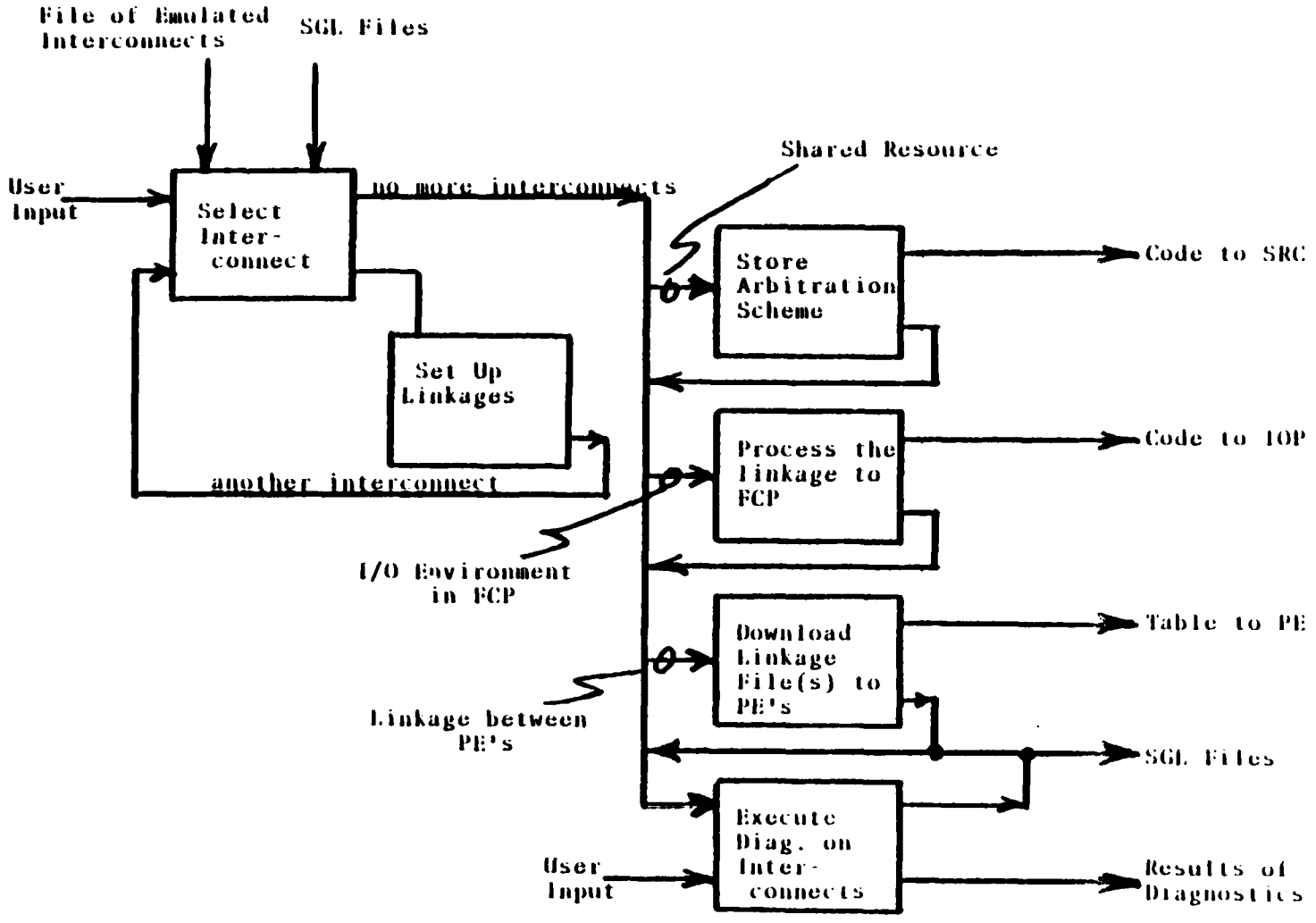
At this point, the emulations of the hardware and of the instruction set have been selected for the parts of a emulated network, called the TS. The user now selects those interconnectors - as buses, shared resources, etc. He will select an interconnector and will then indicate the ports of the I/O devices which he wishes to be connected to that interconnector. SGL will then update the linkage tables for IOP, for SRC, and for all PEXE's.

Once the user finishes the selection of interconnectors, the parts required to interface the shared resources and the linkage tables for SRC are downloaded to that module. The SRC should have several arbitration schemes identified, and these modules thus downloaded will allow the user to select one without having to program it.

Whether or not the user has a shared resource, he will be able to finish the linkages for his TS (one or more CS's) to some section within the FCP. These linkages can be to a memory file, to a tape or a disk file, to a computer program, or to some device external to FCP. This will involve buffer areas, programs to maintain these buffers (utility programs), interrupt-handlers, linkage tables in FCP, etc. Linkage tables will also have to be downloaded to the IOP.

Some linkages will be direct from PE to PE. This section of the linkage tables will be sent to IM within the PE so that direct accesses can be made. Again, the complexity of such accesses will be user-dependent.

It is suggested that diagnostic programs be available to test all choices for interconnectors. Again, it is an user option.



E2B4 Link in Interconnects

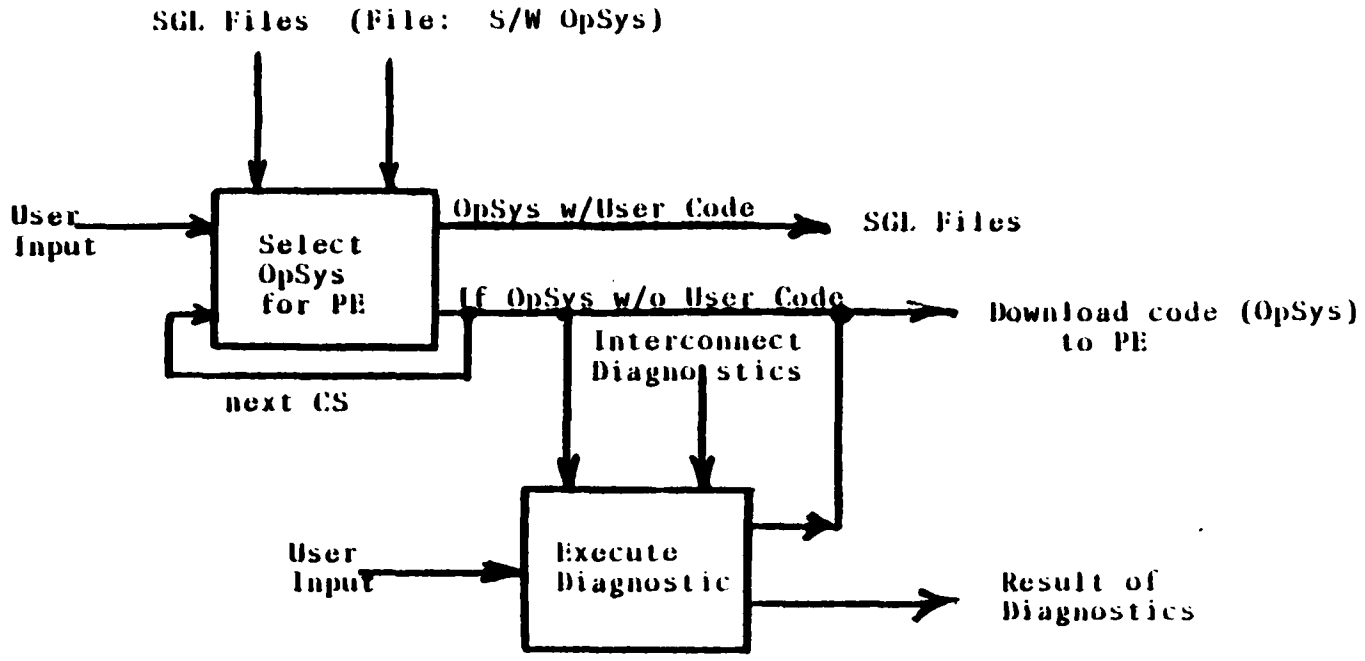
E1C - SELECT TS OPERATING SYSTEM

So far, the emulated hardware modules have been downloaded to the PE's and all linkage has been done so that the TS represents a network of computer systems. The linkage tables (SGL tables) have been saved to enable software interfacing.

If it is possible, in the future, to generalize the various operating systems so that an operating system absolute module can be loaded into Local Memory apart from the absolute modules of the TS application programs, then here is where the absolute operating system would be processed. So far, the various operating systems employ different approaches to loading programs and doing dynamic address translation and paging. Virtual memory techniques and philosophies vary. Since hardware emulators can be designed, so also can software operating systems be generalized and incorporated with interfaces.

Therefore, when it becomes possible to do so, the operating system for each CS (Computer System) will be selected by the user. Interfaces and linkage tables will be created for each CS so that the user can select application programs during RTE. A change of application programs would not necessitate the return to SGL, but could be effected during RTE.

Once the Operating Systems have been downloaded to all CS's, then further diagnostics should be run to see that the CS's within the TS can communicate with each other. Of course, the execution of diagnostics is a user choice.



E1C Select TS Operating System

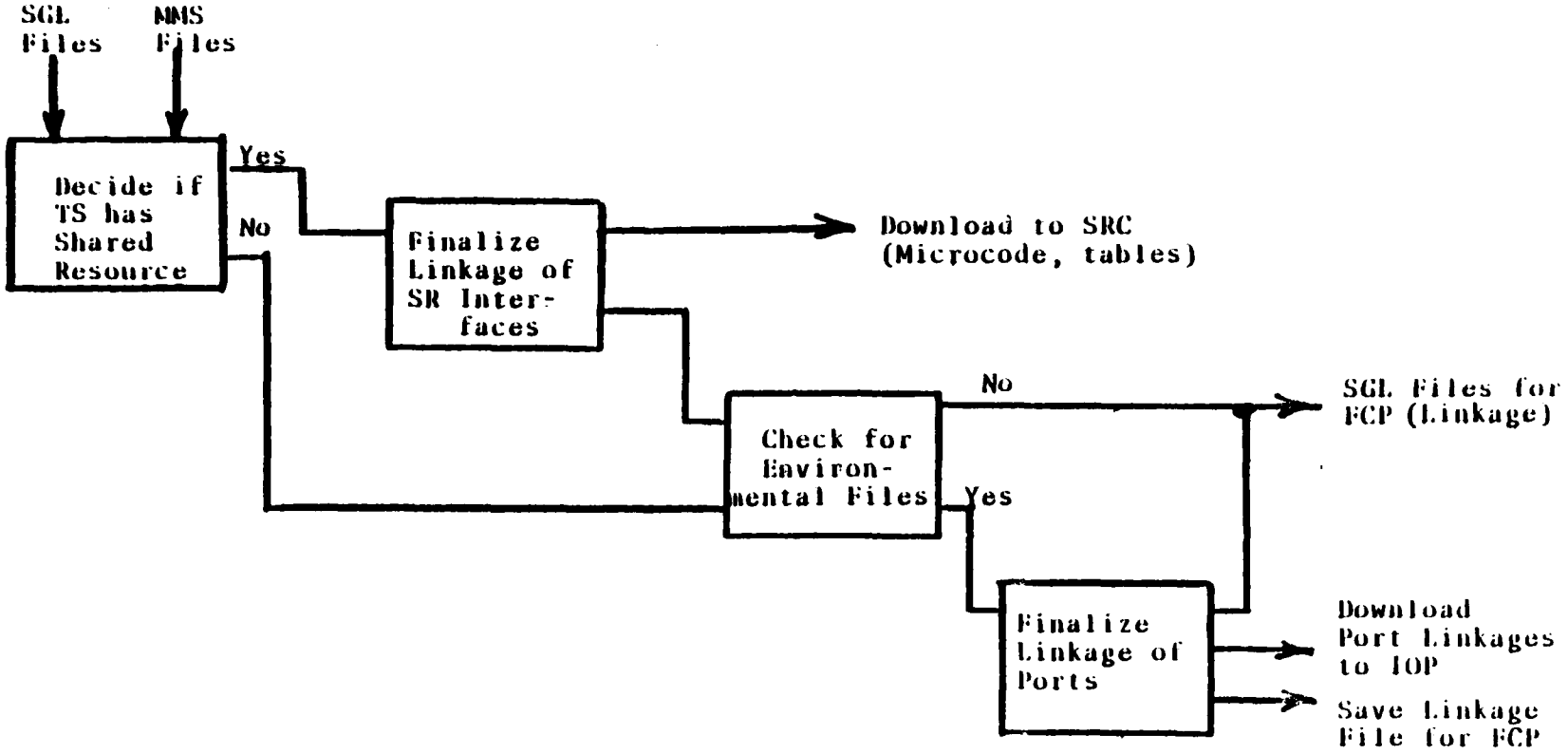
E1D - CONFIGURE "BIG 7"

In this process, the modules existing on the Master Segment are prepared. If the user has specified that a shared resource is part of the TS and that the shared resource requires some arbitration module, then SRC needs to be utilized.

The modules to go to CS of SRC are linked with the executive of the SRC, linkage tables and data are prepared to go to IM of SRC, and all parts are downloaded to SRC.

If the user is connecting the TS to programs within FCP or to some emulated I/O devices within FCP, he has been building the I/O Port Table for IOP. Now the linkage is made to the executive of IOP, and the IOPT (I/O Port Table) is downloaded to the IM of IOP.

FCP retains the necessary linkage tables required by RTE (Run-time Executive).



E1D Configure Big 7

16.1 Abbreviations Used in Section 16

CRT	Cathode Ray Tube
CS	Control Store
FCP	Facility Control Processor
H/W	Hardware
IEC	Instruction Emulation Code
IM	Internal Memory
I/O	Input/Output
IOP	Input/Output Interface; Input/Output Processor
IOPT	I/O Port Table
LM	Local Memory
MMS	Multiple Microprocessor System
Oper.Sys	Operating System
Op.Sys	Operating System
OS	Operating System
PE	Physical Element
PEXE	Executive for PE
PMS	Performance Monitor System
PMSI	PMS Interface
RTE	Run-time Executive
SGL	System Generation Loader
SR	Shared Resource
SRC	Shared Resource Controller
TS	Target System

17.0 RUN-TIME EXECUTIVE

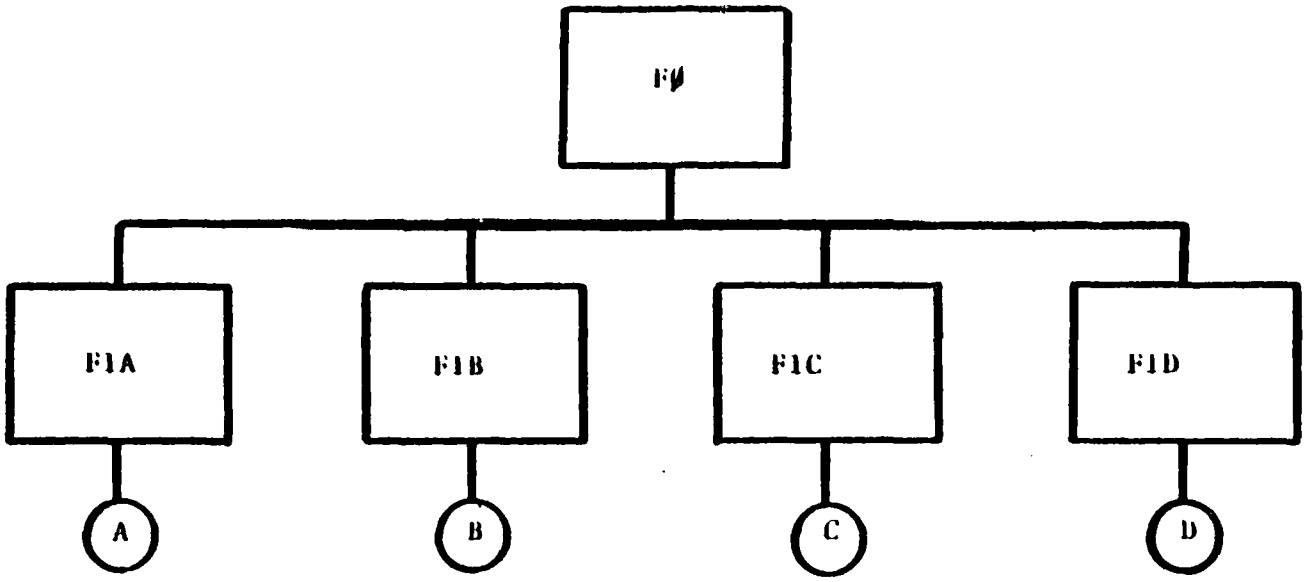
The second package which the MMS general user will call is the Run-Time Executive (RTE). The user has already called System Generation Loader (SGL) to enable him to configure the hardware (and possibly the TS Operating System) of the TS. Linkage Tables exist in the FCP computer which reflect where all the parts of the TS now reside.

During RTE, the user will link in the necessary software and environmental data/programs which allow the execution of the TS. During execution of the TS, the user can obtain results from his application programs, can do DEBUG processing on his program, or can collect performance information on the application program.

The RTE is a large, powerful executive which will accomplish much for the user. It allows the user to execute different application programs on the configured TS without having to re-execute SGL. The linkage tables which are saved from SGL will allow the user to change only the application programs for the TS.

The section "User Scenario" will discuss the usage of RTE from the user viewpoint. RTE will be programmed to be run interactively from a CRT or to be directed from a control file. This last option is beneficial for setting the run to execute during the night hours, especially for those runs that need a lot of time and need much PMS collection.

Figures 17.0-1 a-e are the representation of the Software description which comprises the rest of this section.



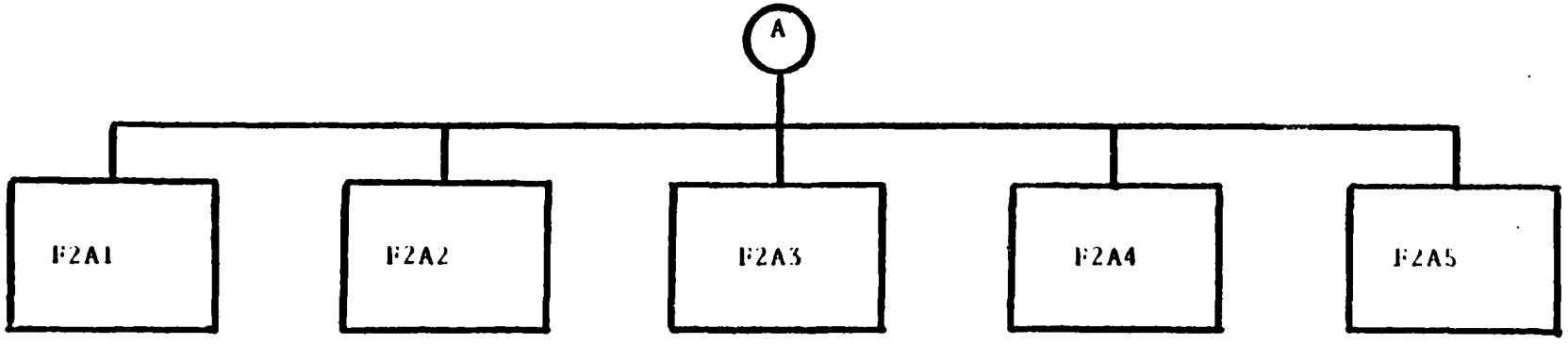


Figure 17.0-1 a,b Top-Down Design Charts

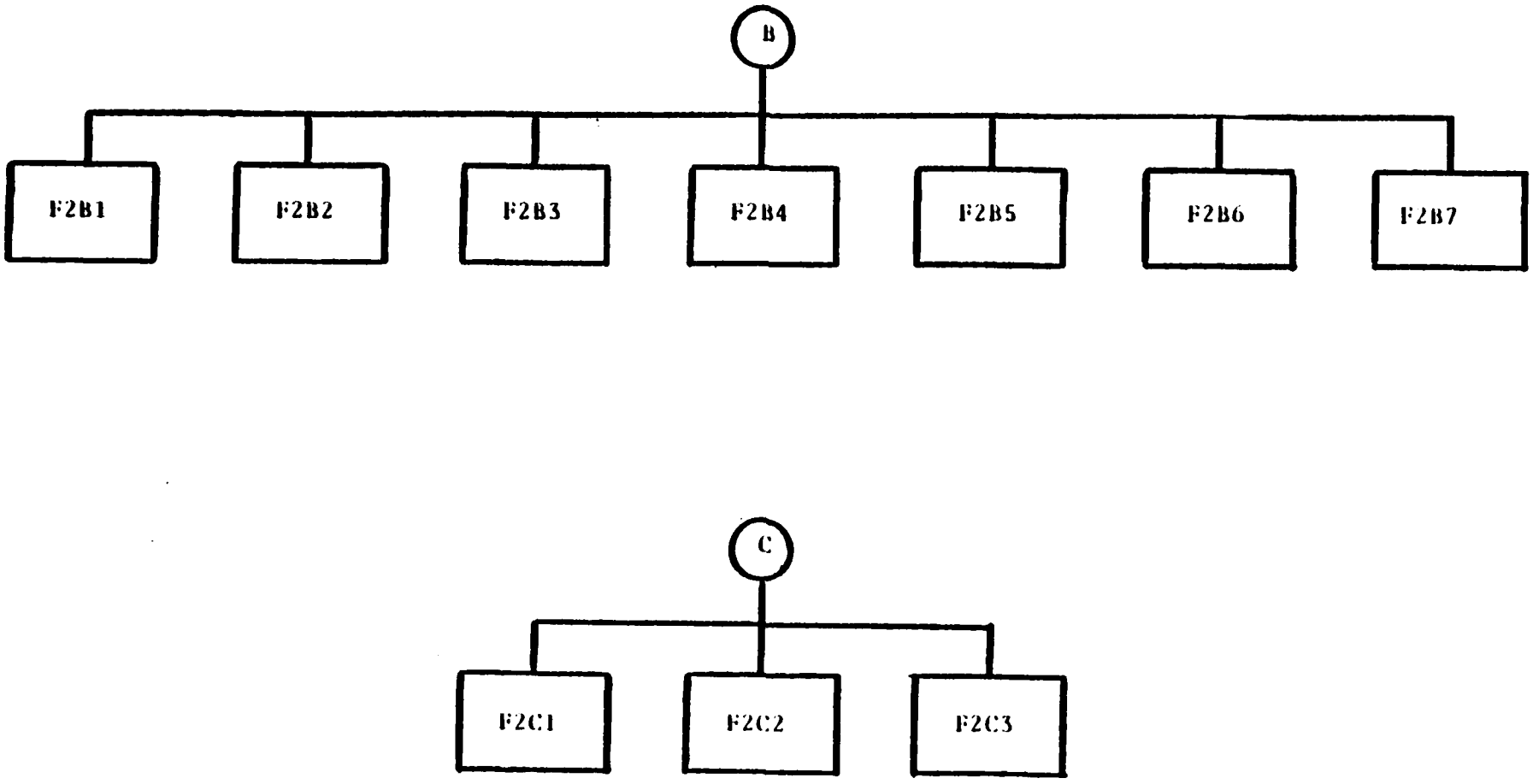


Figure 17.0 1 c,d Top-Down Design Charts

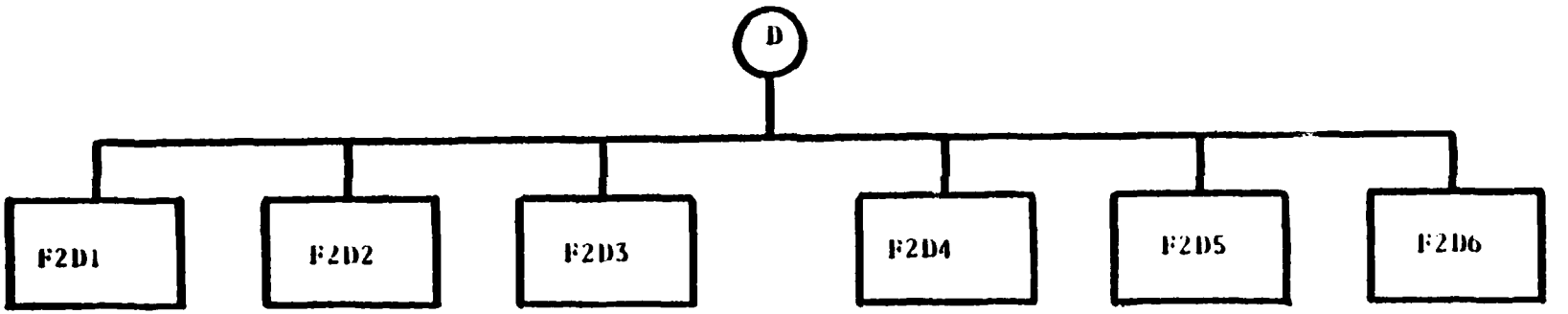


Figure 17.0-1 e Top-Design Charts

FØ - RUN-TIME EXECUTIVE (RTE)

The Run-Time Executive (RTE) supervises all facets of executing the user's Target System (TS). The Run-Time Executive should be initiated by the user immediately upon completion of the System Generation Loader (SGL) phase. The SGL initializes and readies the Target System for execution. The RTE completes the definition of TS I/O; optionally, configures user controlled debugging or performance monitoring, and then executes the Target System.

The Run-Time Executive software is broken into four phases. The first phase, "Configure Environment" (F1A), completes the definition of all Target System I/O by defining buffer memory, defining the handler software, and updating tables within the hardware. The "Configure Debug" (F1B), allows the user to define debugging options for computers within his Target System. Rather than choosing to DEBUG his Target System, the user may execute under the Performance Monitoring System (PMS). This third phase of RTE, "Configure PMS" (F1C), allows the user to select his options for the Performance Monitoring System. The "Execute TS" (F1D) allows the user to evaluate his emulated system through the interactive use of either DEBUG or PMS.

The Run-Time Executive utilizes two major sets to control and communicate with its phases.

The RTE I/O descriptor (see figure 17.0-2) is initiated by SGL and is completed by RTE. It contains all information necessary to set up and supervise the operation of the Target System I/O. This data set contains a record for each I/O emulator in the Target System. This record is a duplicate of the I/O Port Table contained in the I/O Processor (IOP). In addition, this record contains identifying names of the I/O for use in communicating with the user.

The second RTE data set is the DEBUG/PMS Option Table Format. This data set enables RTE to control the operation of the Target System under DEBUG or PMS. This data set contains a set of records for each computer within the TS. This set of records is depicted in Figure 17.0-3.

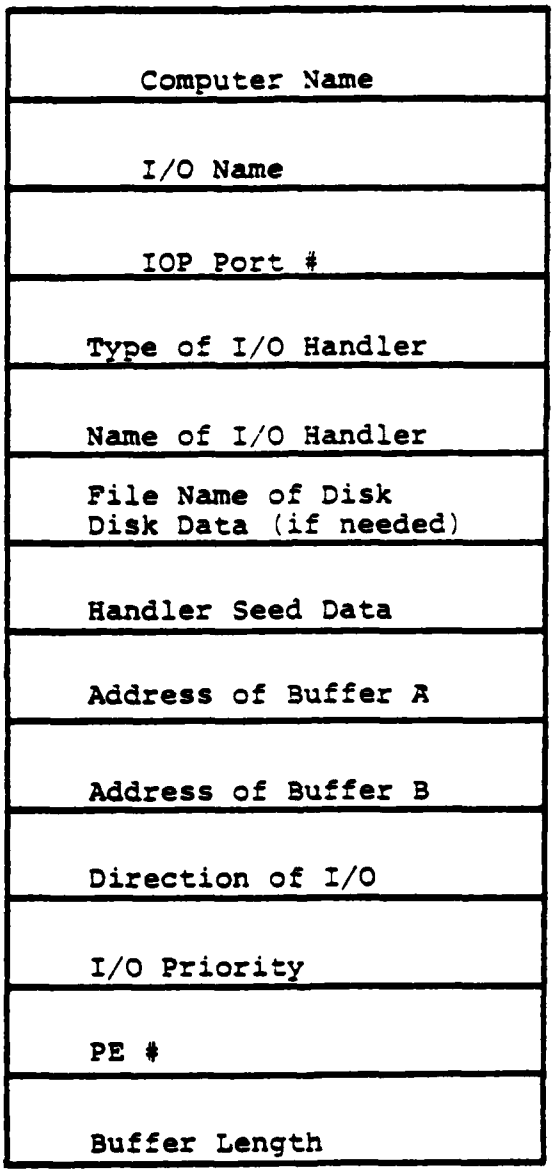


Figure 17.0-2 RTE I/O Descriptor Record Format

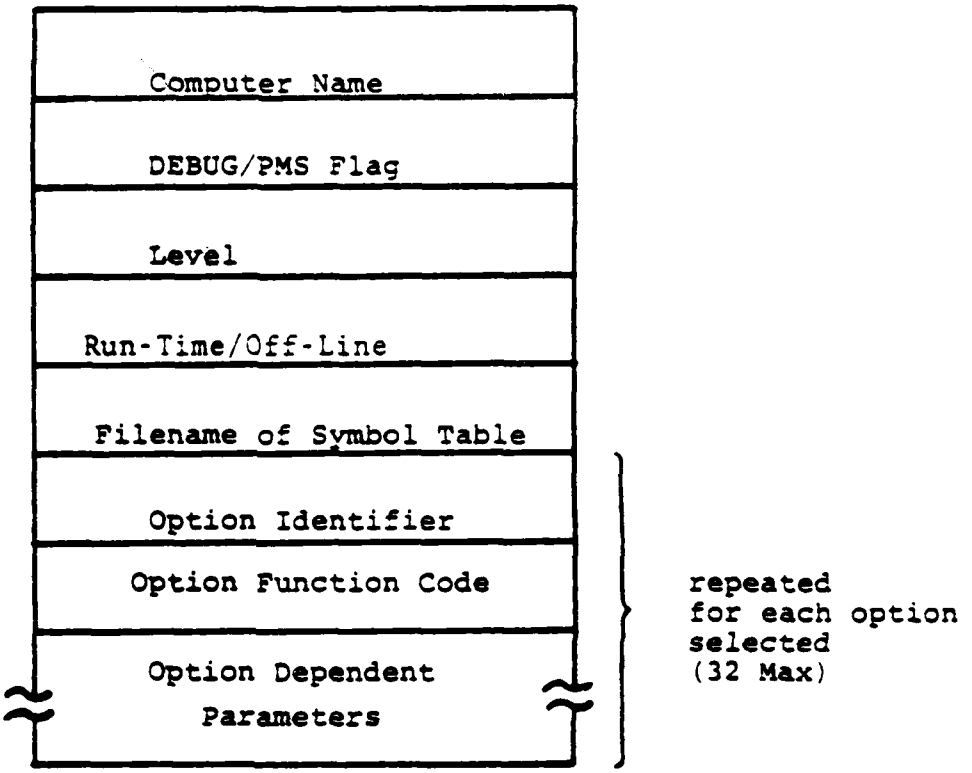
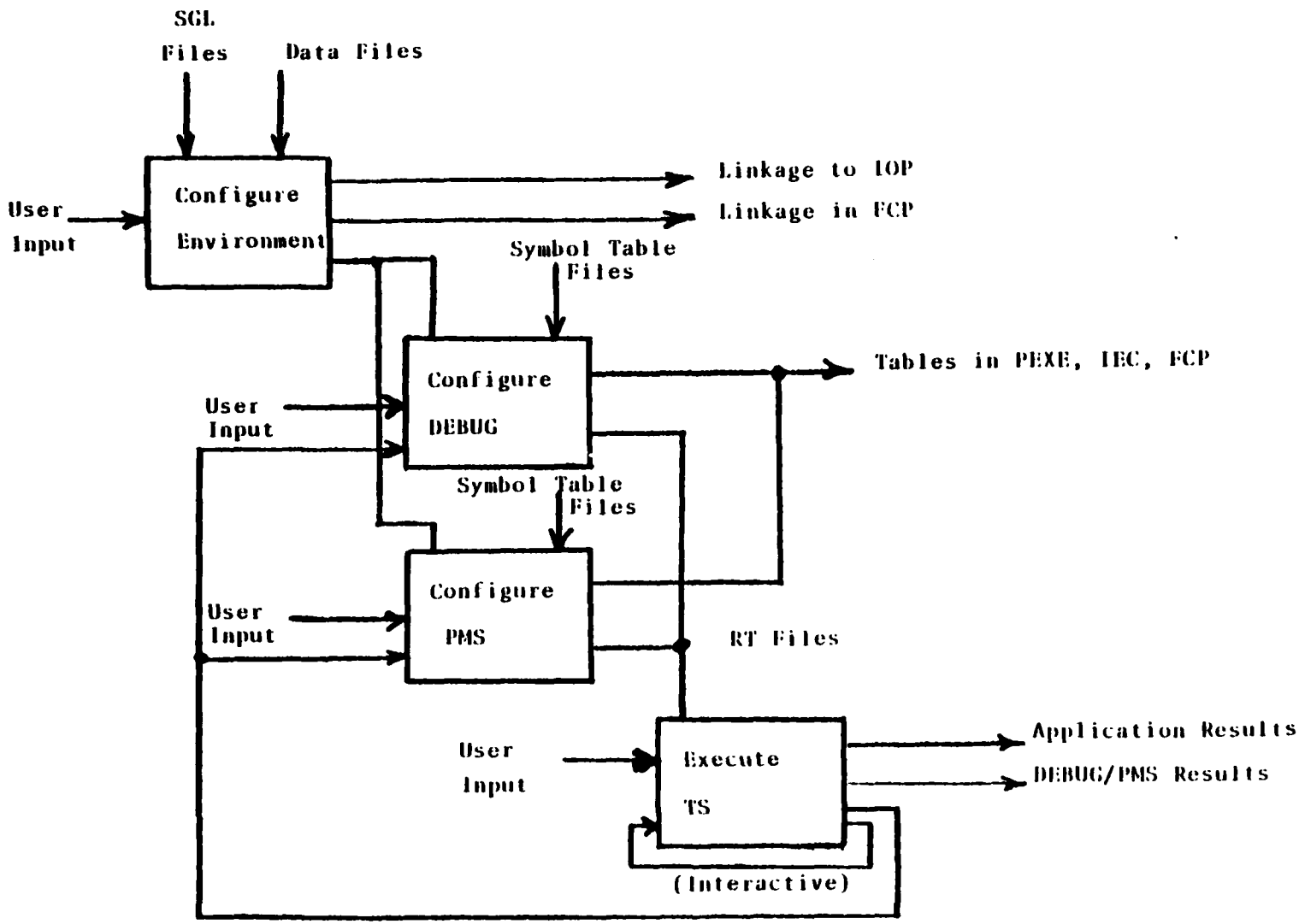


Figure 17.0-3

RTE DEBUG/PMS OPTION TABLE FORMAT



F Ø Run-Time Executive (RTE)

F1A - CONFIGURE ENVIRONMENT

This initial phase of the RTE completes the definition of the Target System I/O process. These definitions consist of configuring the I/O support environment for the Target System within the Facility Control Processor (FCP) computer. For each I/O in the Target System, the user is asked to define:

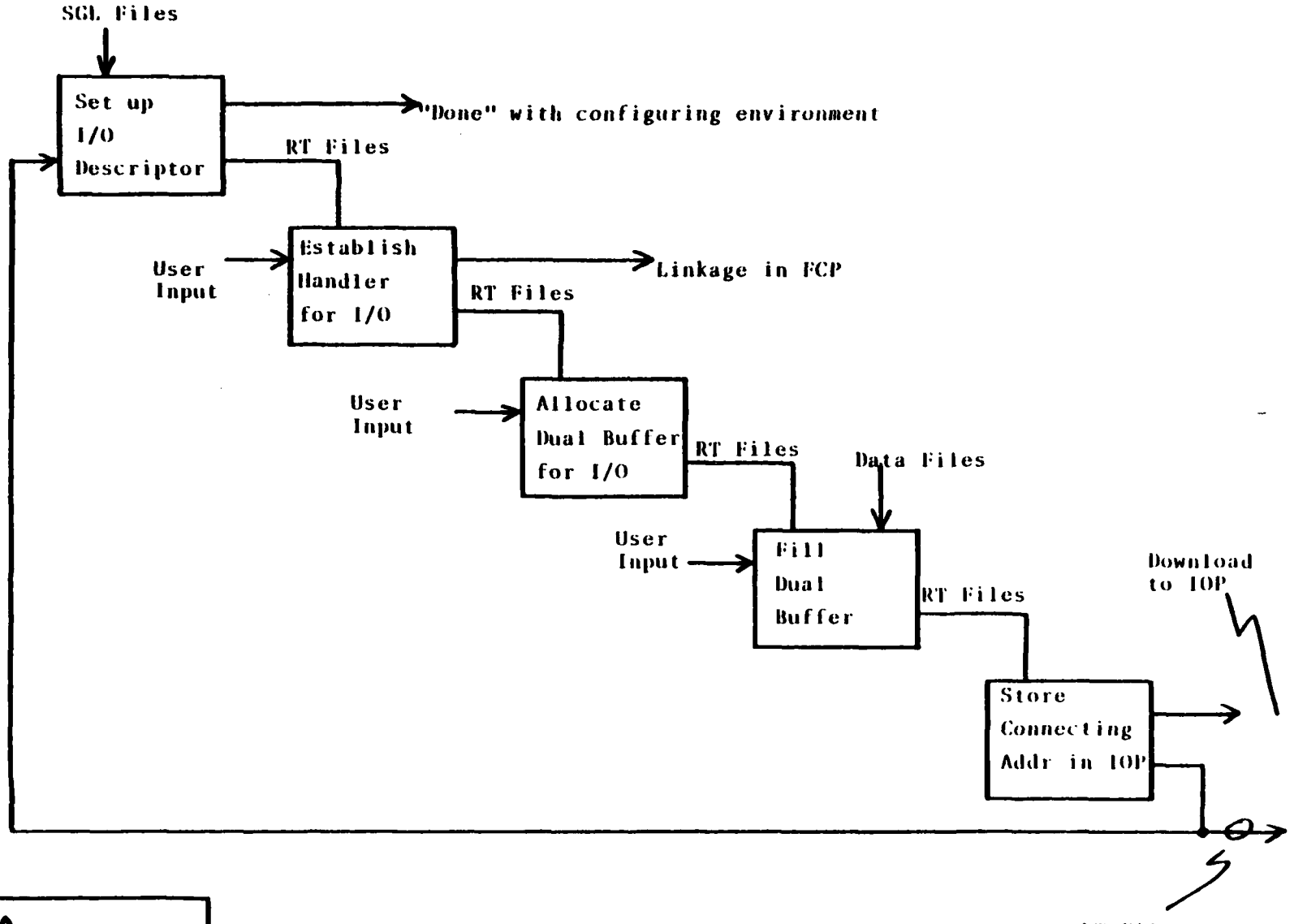
a) a Handler for the I/O

b) a disk file, if appropriate

c) the I/O buffer size

d) the initial "SEED" data for generating input data to a PE

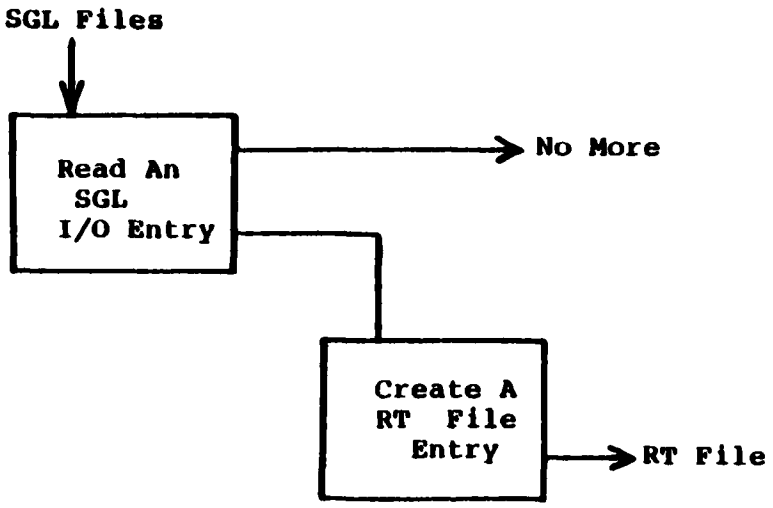
Delaying the definition of this information until run-time allows the user the flexibility of re-configuring his support environment (i.e., FCP resources) without the necessity of reconfiguring his Target System. Thus, if a particular I/O support scheme is not as responsive as the user desires, he may re-configure that scheme to improve that response.



F1A Configure Environment

F2A1 - SET UP I/O DESCRIPTOR

This step, "Set Up I/O Descriptor" (F2A1), begins the environment configuration RTE phase. This step reads an IOPT entry as set up by the SGL phase. If no entry is present, the environment configuration is finished. If an entry is present, this step creates an I/O control record in the Run-Time files.



F2A1

Set Up I/O Descriptor

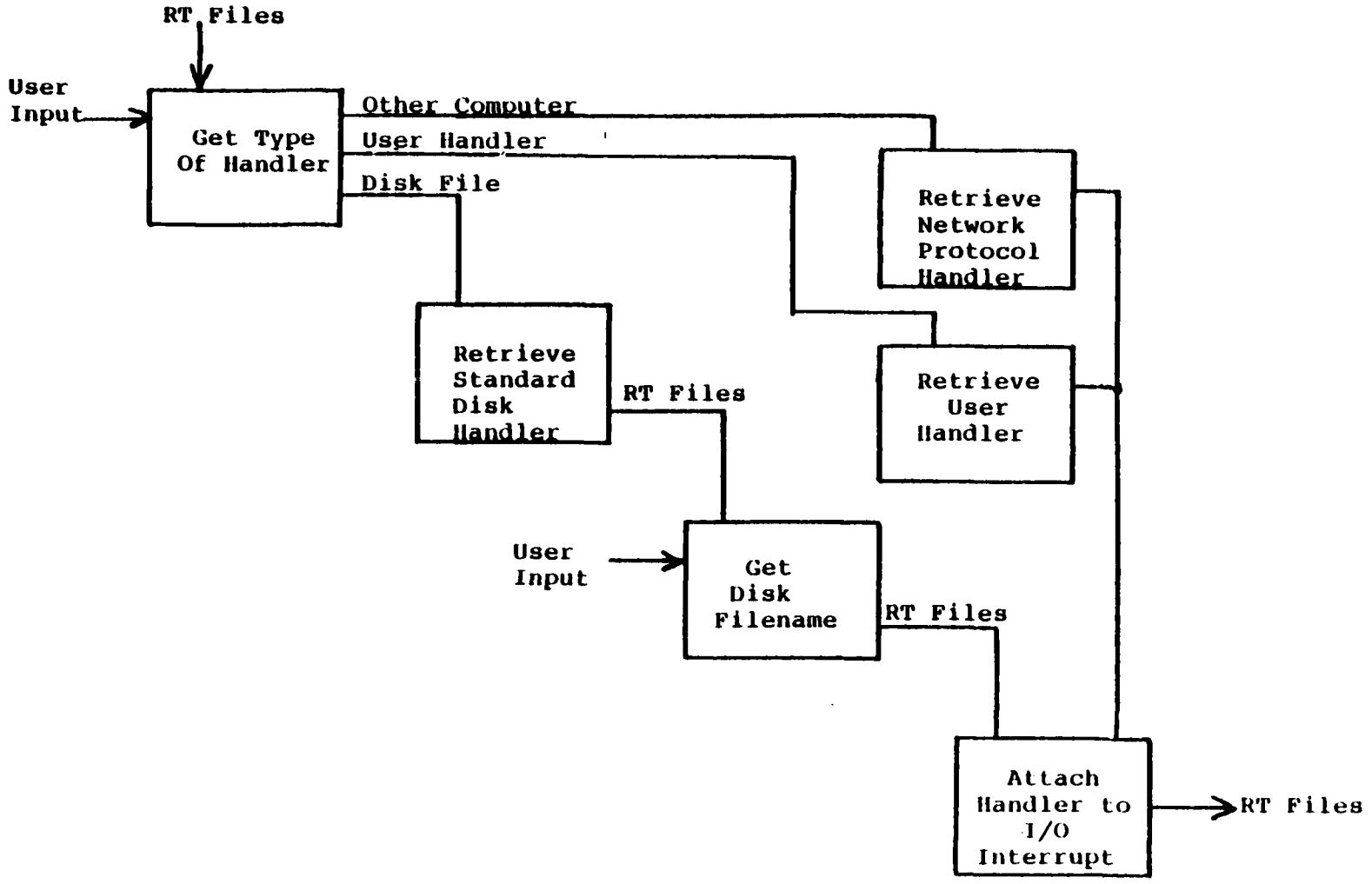
F2A2 - ESTABLISH HANDLER FOR I/O

One of three types of handlers may be selected for any I/O. These types are:

- An existing disk file within the FCP.
- An existing program within the FCP.
- An external data processor whose network interface has been defined within the FCP.

Any of these types may either produce or consume data as required by the defined Target System environment.

This step, "Establish Handler for I/O" (F2A2), queries the user for the type of his desired handler. If the user selects a disk file, the user is queried for a filename. The vendor-supplied disk handler, with appropriate linkage to the user's filename, is then utilized as the I/O handler. If the user selects a program handler type, he is queried for a program name. If the user selects an external computer type, then he must enter the name of the routine which handles interfacing with his desired computer. Finally, the user's selected handler is linked to the port table, IOPT, which resides within IOP.

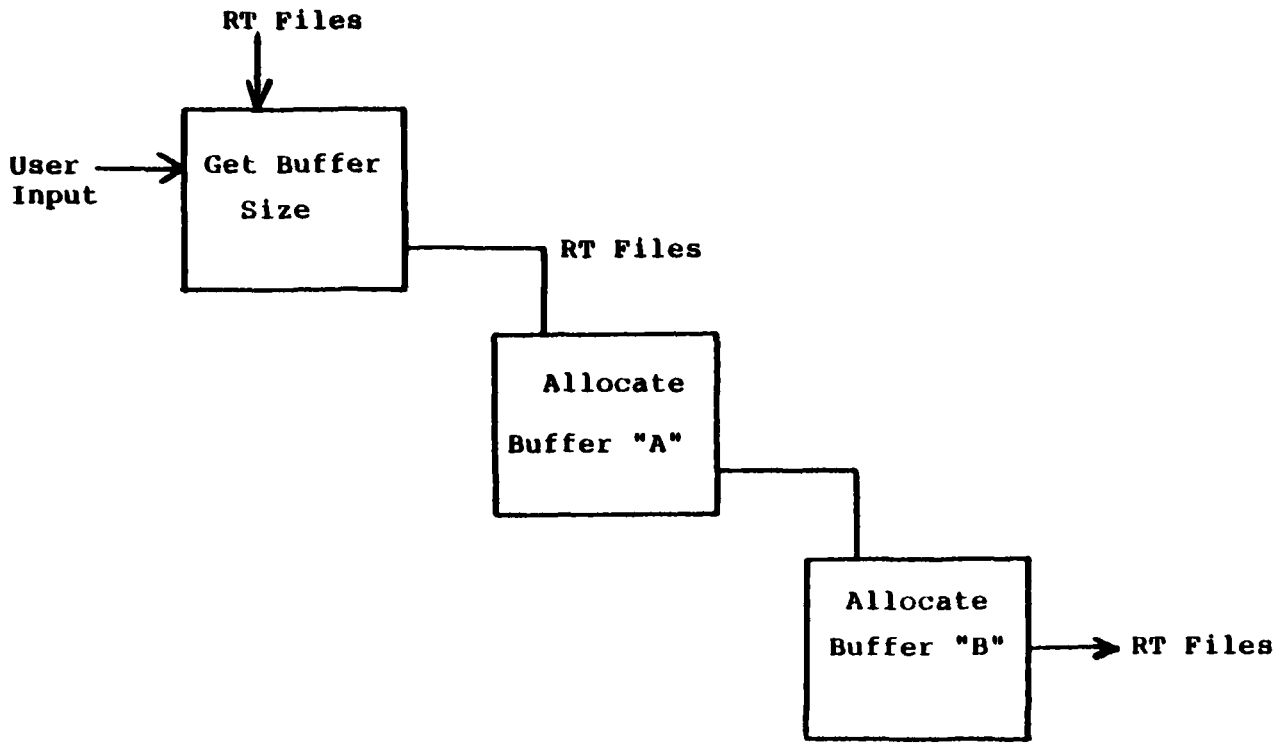


F2A2

Establish Handler for I/O

F2A3 - ALLOCATE DUAL BUFFER FOR I/O

This step, "Allocate Dual Buffer for I/O" (F2A3), allocates the memory resource within the FCP to support a particular I/O. It is invoked for each IOPT entry defined during SGL. If the "filler" type is a disk file, then the sector size of the FCP mass storage media is used, thus minimizing unnecessary user interaction. For non-disk "filler" types, the user is allowed to select a buffer size which is optimal for this, more user intensive, I/O.



F2A3

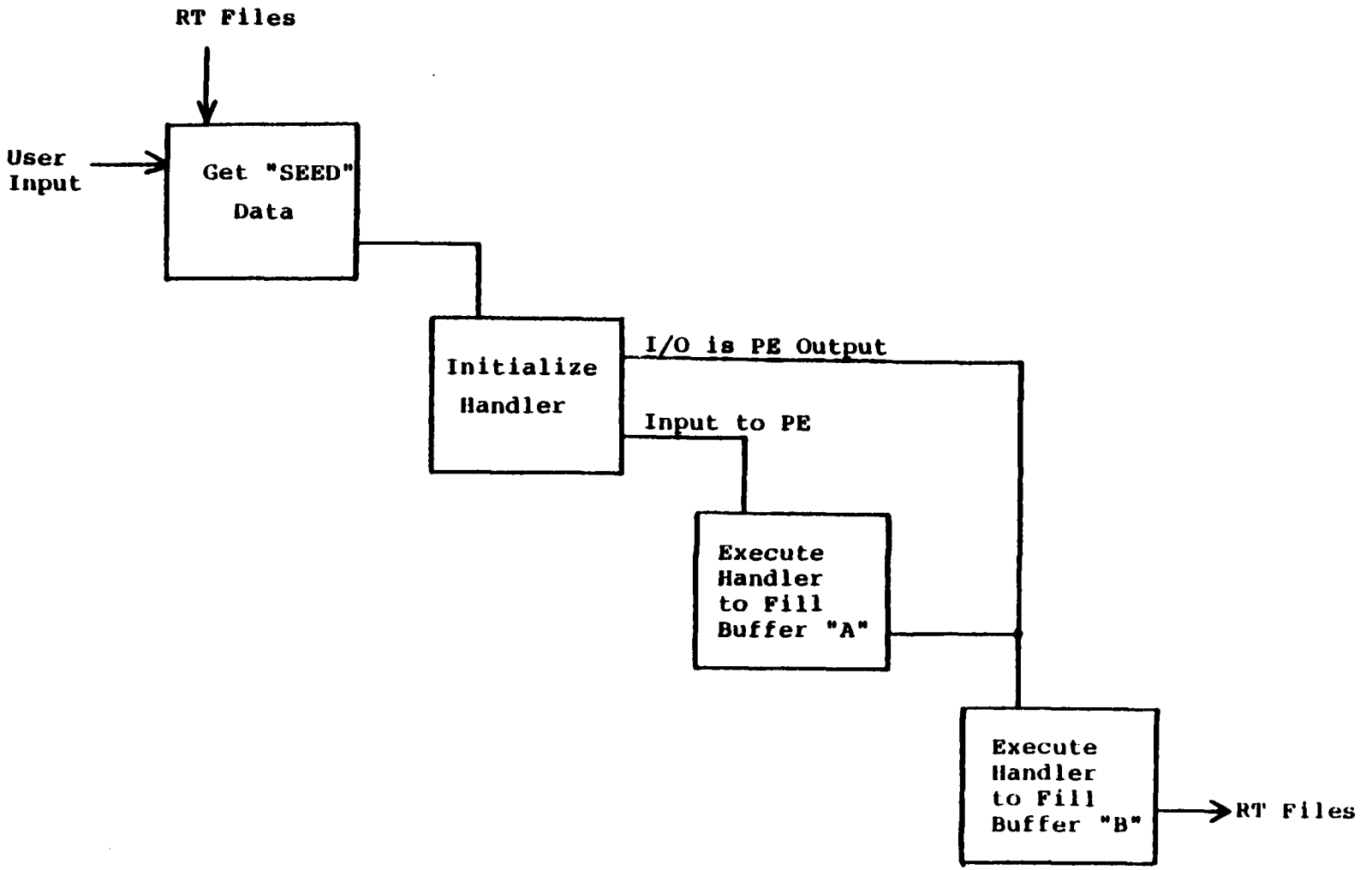
Allocate Dual Buffer for I/O

F2A4 - FILL DUAL BUFFER

For I/O types that involve input to the PE from FCP, the user enters a "seed" data item. This allows randomizing of Target System inputs. The I/O handler is always given the current seed. The Handler may then apply any computer program of its choice in generating data to be input to the PE and in creating the new seed. This new seed is stored by RTE for the next call of the handler.

The vendor-supplied disk handler need not utilize this seed data item. Rather, it will sequentially read the user-specified disk file. This disk handler need only read sequential disk sectors and present them to the IOP as input to the PE. The seed data item should be used, however, as temporary storage for the disk sector address. This allows the disk handler to be reentrant, as it must be capable of handling many separate PE inputs.

This step, "Fill Dual Buffer" (F2A4), performs four functions. First, the seed data item is obtained from the user. The I/O handler is initialized. For PE input which emulates a type as disk or tape input data, the handler is then executed to fill each of the dual buffers.

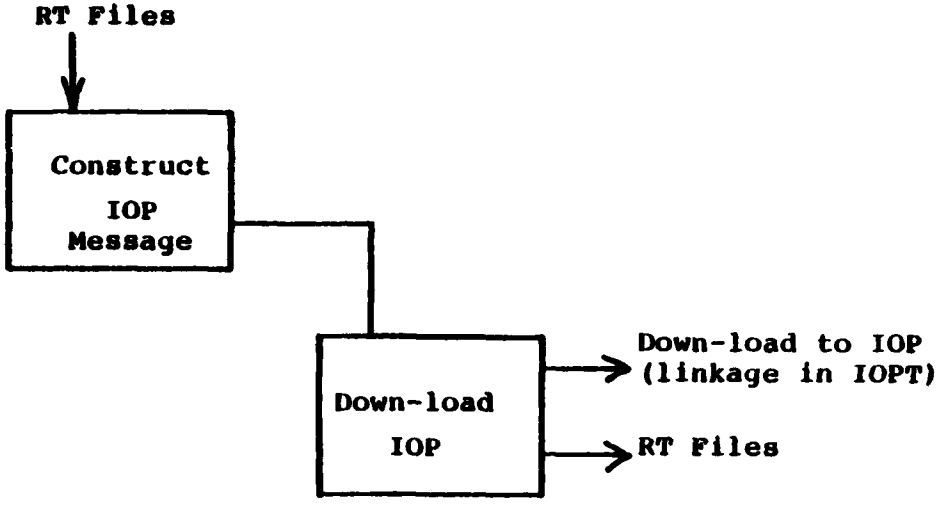


F2A4

Fill Dual Buffer

F2A5 - STORE CONNECTING ADDRESS IN IOP

This step, "Store Connecting Address in IOP" (F2A5), completes the configuring of a particular I/O. The "Configure Environment" phase (F1A) of RTE will now loop to process the next I/O defined by the SGL phase. Once all linkages have been completed, the completed IOPT table is downloaded to IOP.



F2A5

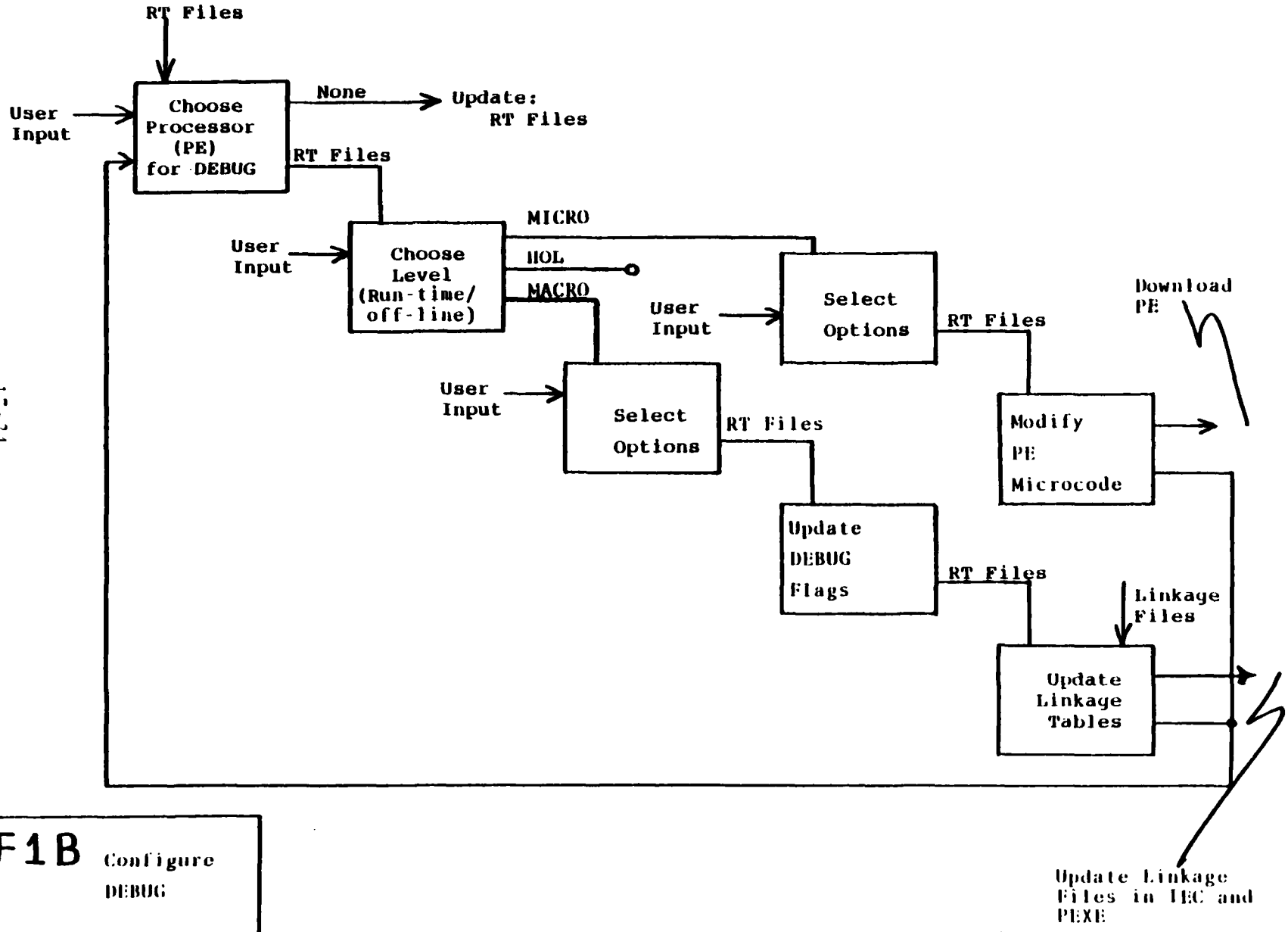
Store Connecting Address in I/O

F1B - CONFIGURE DEBUG

The "Configure Debug" (FIB) phase is the second of four phases of the Run-Time Executive. This phase is highly interactive. It allows the user a powerful means for analyzing and debugging his Target System. The user may select for debugging any, or all, of the PE's in his configured Target System. Further, the user may choose to debug his emulator (microprogram level) or his application (macroprogram level) for any given PE. For future enhancement to MMS, the user may choose to do DEBUG on a HOL, but this feature is not currently part of the MMS system.

The user scenario is defined in detail in Section 21. Every effort was made in the design of this scenario to satisfy the needs of all levels of users. The naive user is guided through his various option selections by extensive "help" displays. The more sophisticated user may by-pass these "help" displays and rapidly enter his options by concatenating several selections together. Queries of the user and his responses must always be recorded on a disk file. This will allow the user to print this response file should any questions as to his option selection arise at some later date.

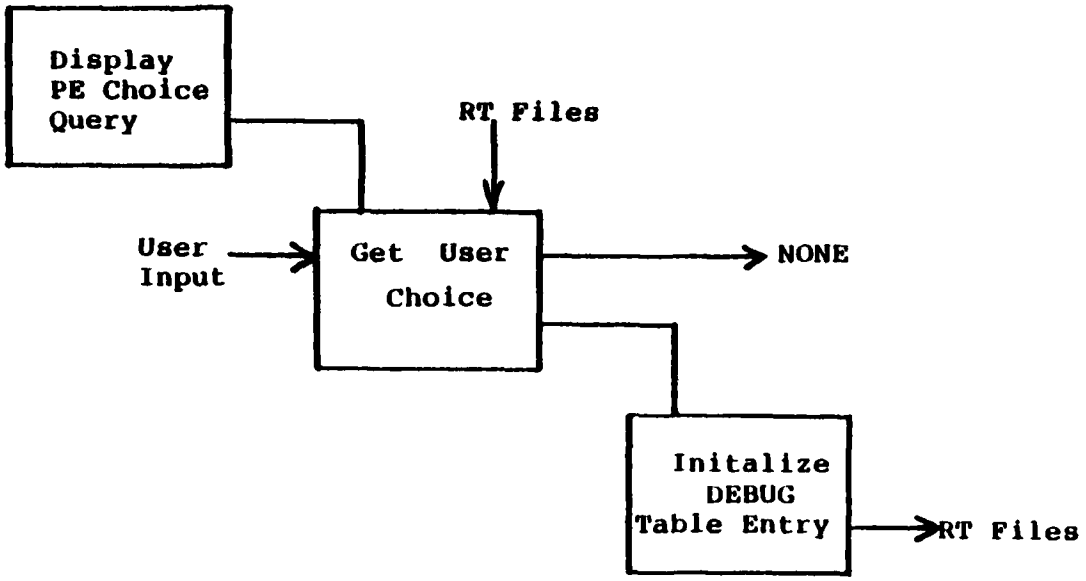
This phase has been divided into seven separate steps.



F1B Configure DEBUG

F2B1 - CHOOSE PROCESSOR (PE) FOR DEBUG

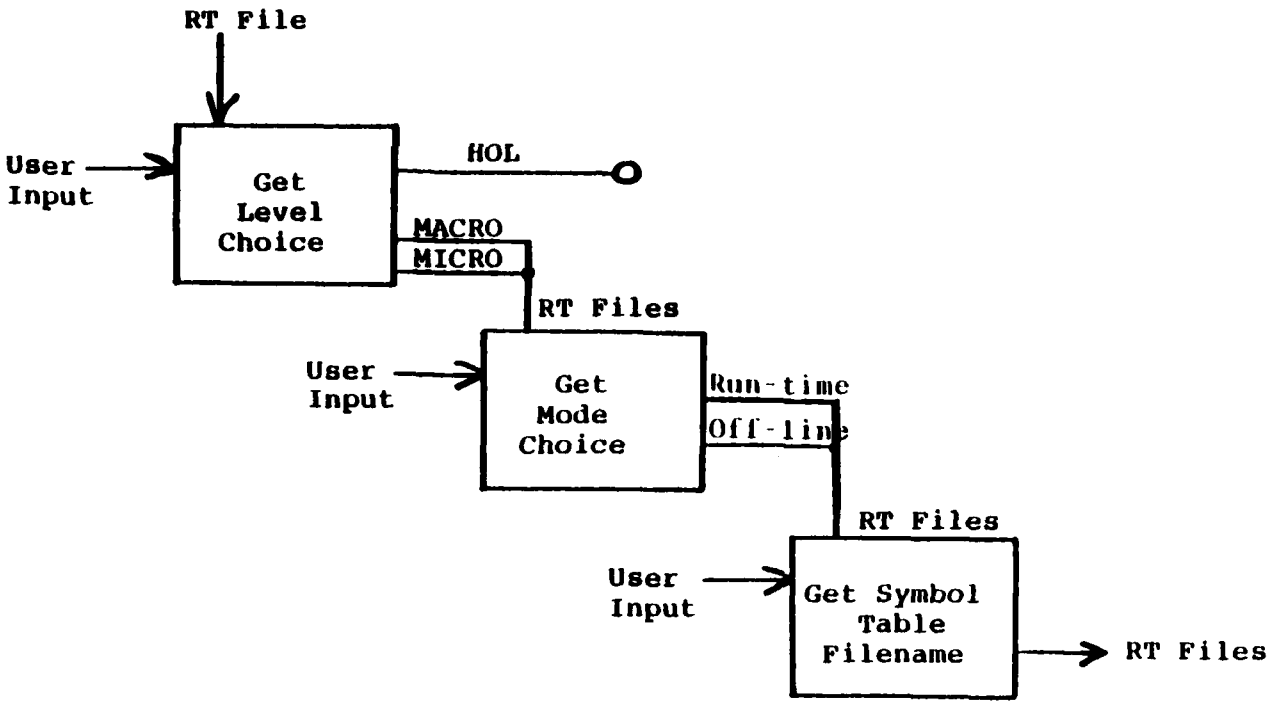
This step, "Choose Processor (PE) for Debug" (F2B1), is the first step of the DEBUG configuration phase. This step allows the user to select a PE configured in his Target System for debugging. The user should be allowed to make this selection in terms of computers being emulated. This allows the user to think in logical terms rather than having to remember the physical emulation components (PE's) of his Target System.



F2B1
Choose Processor
(PE) for DEBUG

F2B2 - CHOOSE LEVEL

Having selected a computer to be debugged, the user is queried for the parameters of his DEBUG run. This second step of DEBUG configuration, "Choose Level" (F2B2), asks three questions of the user. The user must select the level of debugging. Three choices are possible: micro-, macro-, or HOL-. The HOL-level is not implemented at this time; however, stubs are provided for HOL- level debugging should this capability be added in the future. The user must select either run-time or off-line mode of debugging. In the off-line mode, the RTE execution phase will store all data collected by DEBUG in a disk file for later analysis. In the run-time mode, the execution phase will display all DEBUG data to the user as it is collected. Finally, the user may select a symbol table file for use in his DEBUG run. DEBUG will use this symbol table to convert numeric addresses and opcodes to symbolic equivalents for presentation to the user. This allows the user to think about his data in logical, symbolic terms rather than in physical terms.



F2B2

Choose Level

F2B3 - SELECT OPTIONS

With this step, "Select Options" (F2B3 or F2B4), the user enters his debugging options. These options are described briefly here and in greater detail in Section 21, "USER SCENARIO". This step presents identical displays to the user for both macro- and micro-level debugging. However, not all options are valid for both levels of debugging. A powerful set of debugging options are available at the macro-level. All macro-level options are assisted by the PE hardware. Thus, these options add little or no overhead to the execution of the TS emulation. The micro-level debug options are not hardware assisted. Therefore, the micro- options are limited to those that can be implemented by a trap to the PE executive software.

In designing the option, selections reflected as part of the user scenario, two requirements were considered. These are the requirement for aiding the naïve user and the requirement for quick and easy use by the sophisticated user. This user scenario provides an optional set of powerful help display which lead the naïve user through each step of the option selection process. The sophisticated user may bypass these help displays and enter his selection simply by concatenating his various selected parameters.

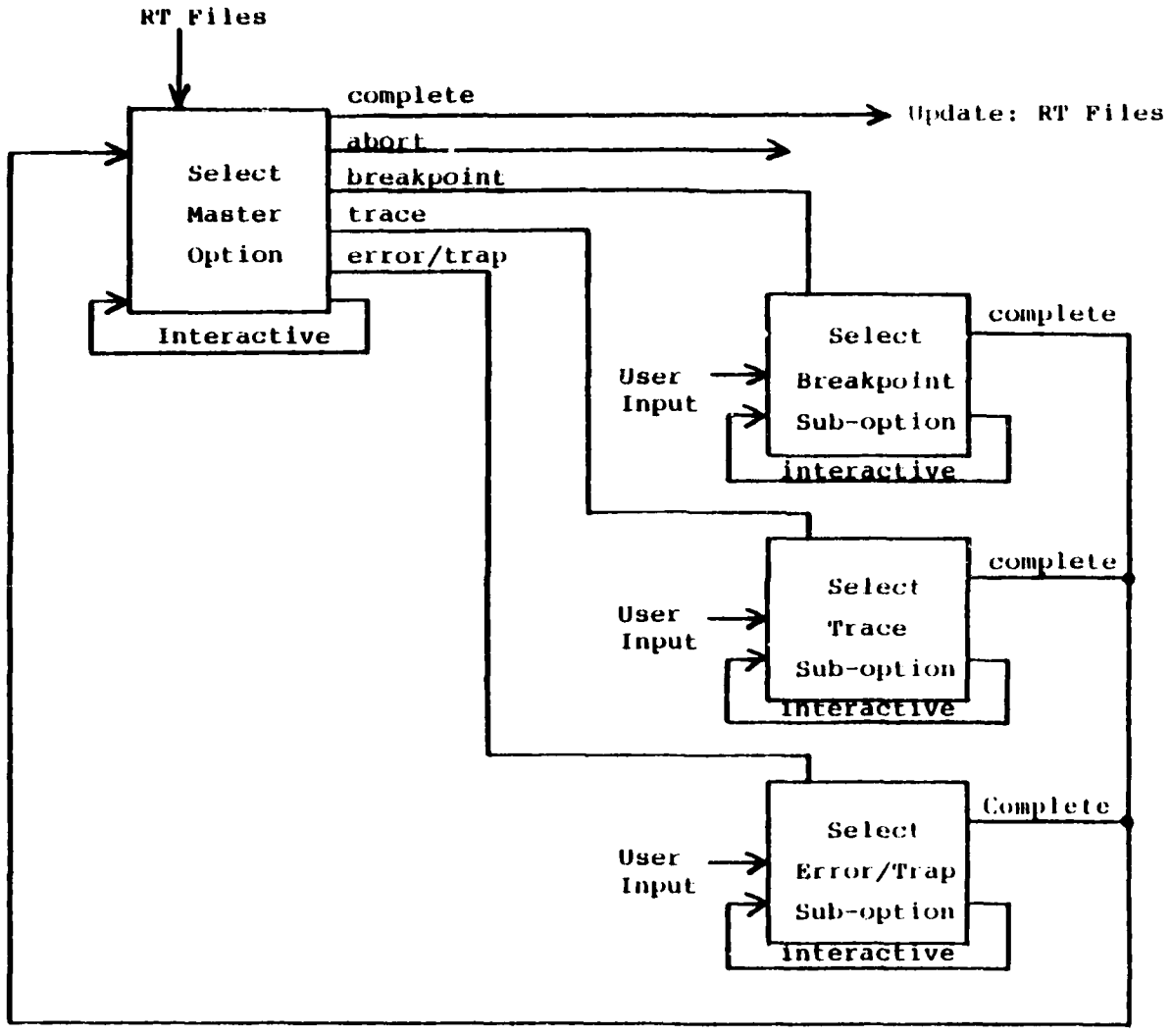
Briefly, macro-level debug options are:

1. Obtain a Help display.
2. Abort debug option selection.
3. Complete debug option selection.

4. Enter quick mode of option selection.
5. Delete a prior option.
6. Set a breakpoint on:
    a. a specific address
    b. a specific instruction opcode
    c. after a specific (pseudo) time interval
7. Select a trace on:
    a. a specific address or range of addresses
    b. a specific instruction opcode
    c. after a specific (pseudo) time interval
8. Trap to a user supplied routine in the FCP.
9. Error stop on:
    a. bus time-out
    b. memory protection fault
    c. execution outside a specific address range
10. Inspect a memory location.

Briefly, micro-level debug options are:

1. Obtain a help display.
2. Abort debug option selection.
3. Complete debug option selection.
4. Enter quick-mode of option selection.
5. Delete an option.
6. Set a breakpoint at an address.
7. Trace an address.
8. Error stop on either bus time-out or memory protection fault.
9. Inspect a memory location.



F2B3/F2B4

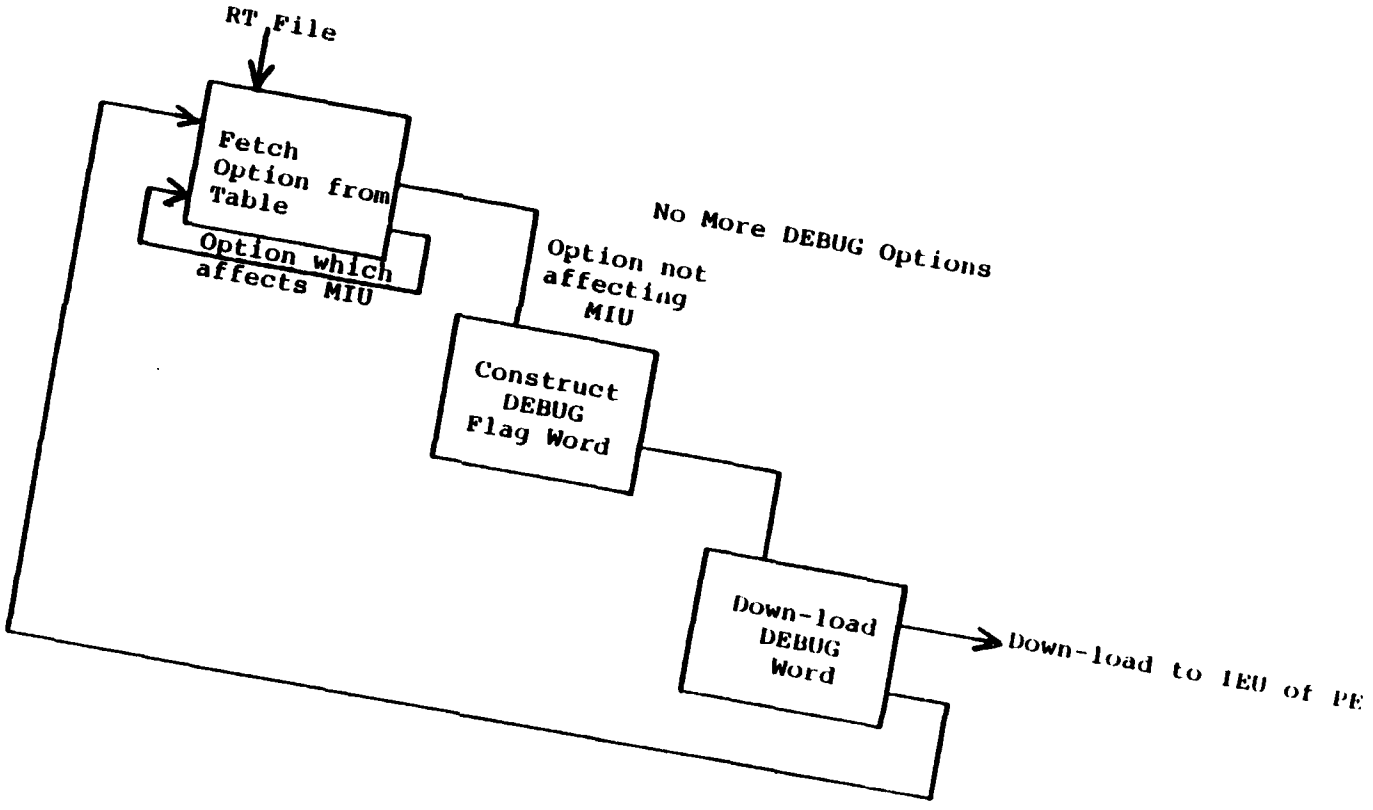
Select Options

F2B4 - SELECT OPTIONS

The description for this process is the same as that for F2B3, "Select Options".

F2B5 - UPDATE DEBUG FLAGS

This step, "Update Debug Flags" (F2B5), is executed for macro-level DEBUG options. This step sets the hardware DEBUG trap flags to options not affecting the Memory Interface Unit. These DEBUG flags will be stored in Internal Memory in the Macro Address Table.

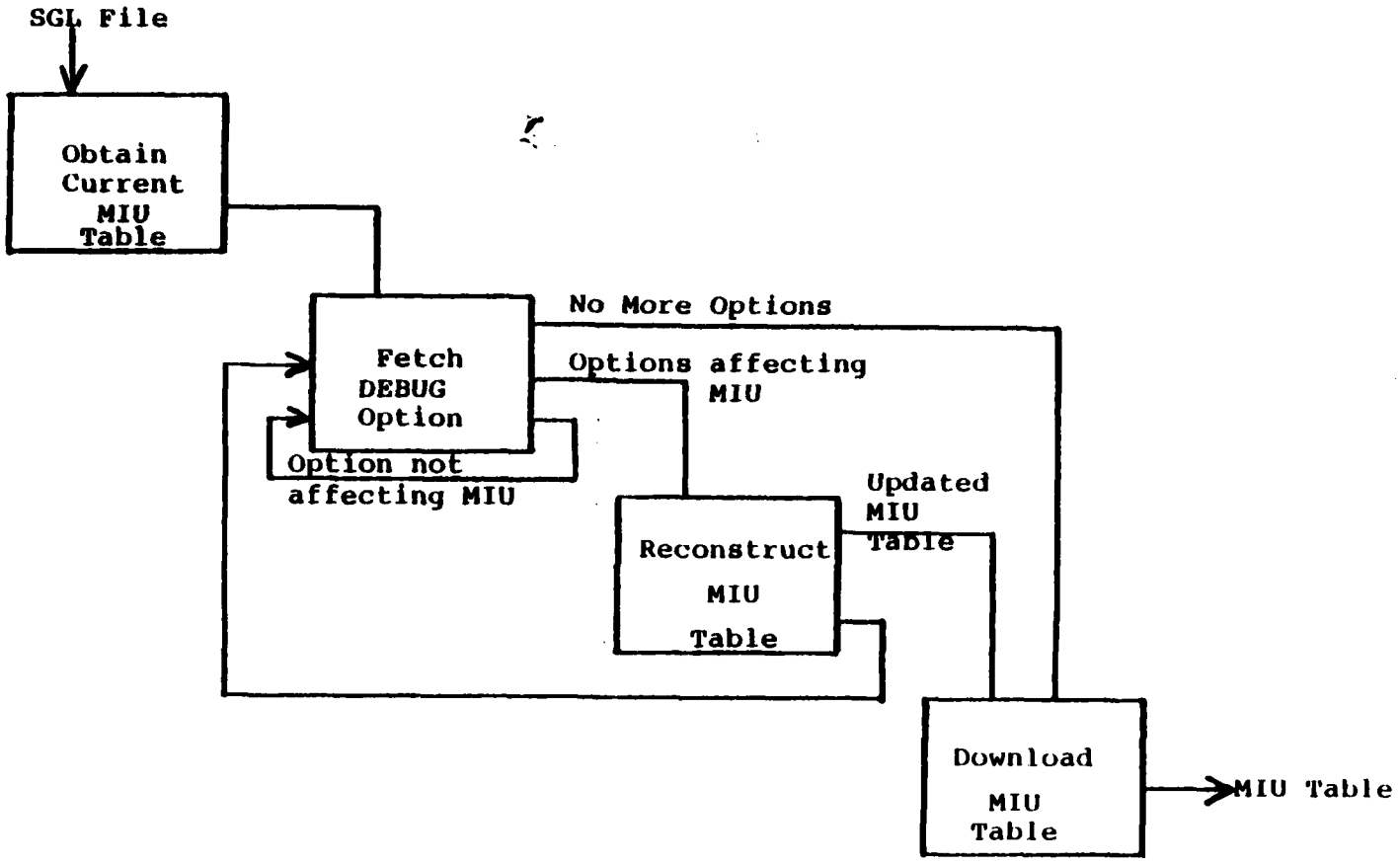


F2B5

Update DEBUG Flags

F2B6 - UPDATE LINKAGE TABLES

This step, "Update Linkage Tables" (F2B6), completes the configuring of macro-level DEBUG options. This step modifies the contents of the Memory Interface Unit. The MIU performs a logical-to-physical address mapping for the PE hardware. A logical block of memory within the emulated Target System is mapped to a physical block within the PE. This table also includes flags for initiating DEBUG whenever the block is referenced. The SGL phase initiates this table as it down-loads the user's Target System software. This step of RTE modifies this table to set up the hardware DEBUG flags to process the user selected DEBUG options.

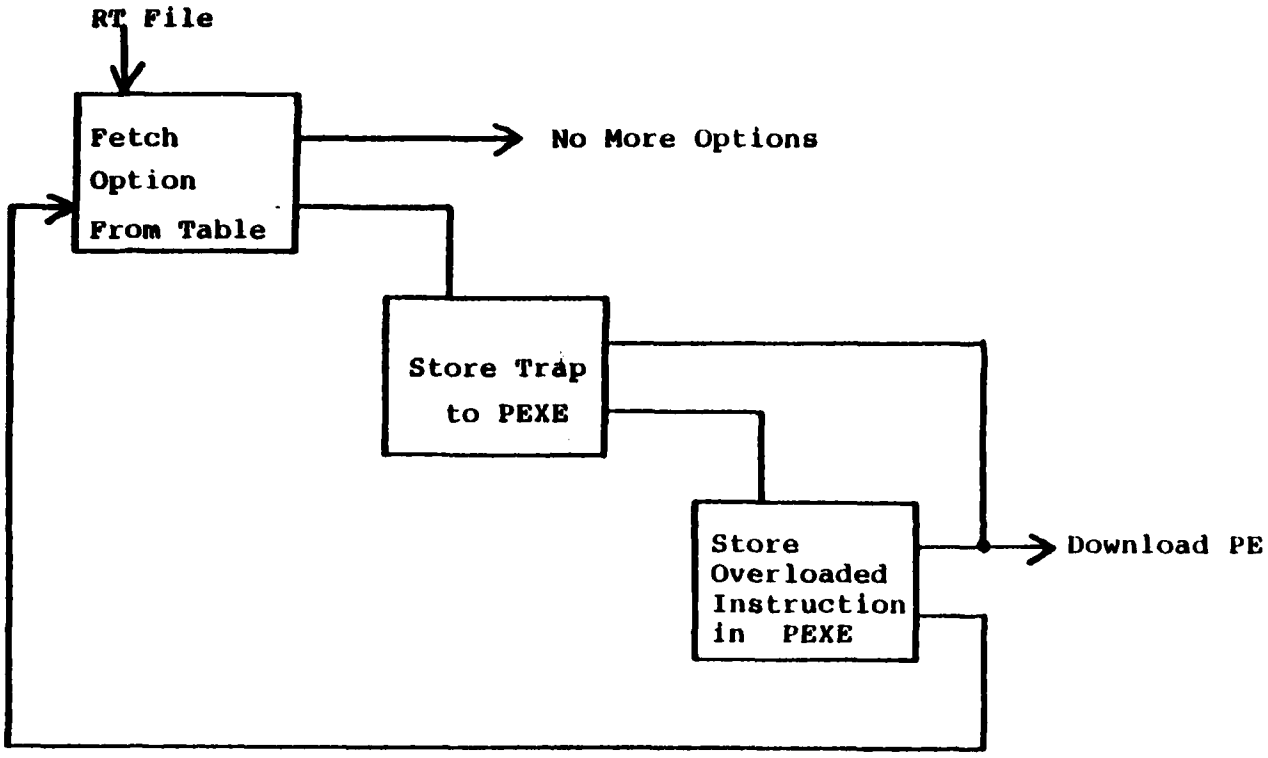


F2B6

Update Linkage Tables

F2B7 - MODIFY PE MICROCODE

This step, "Modify PE Microcode" (F2B7), completes the configuration of micro-level debugging. The PE hardware provides no assistance for micro-level debugging. Thus, each option is implemented as a trap to PEXE. When a micro-level DEBUG trap occurs at run-time, PEXE will interact appropriately with RTE tables within the FCP.



F2B7

Modify Microcode

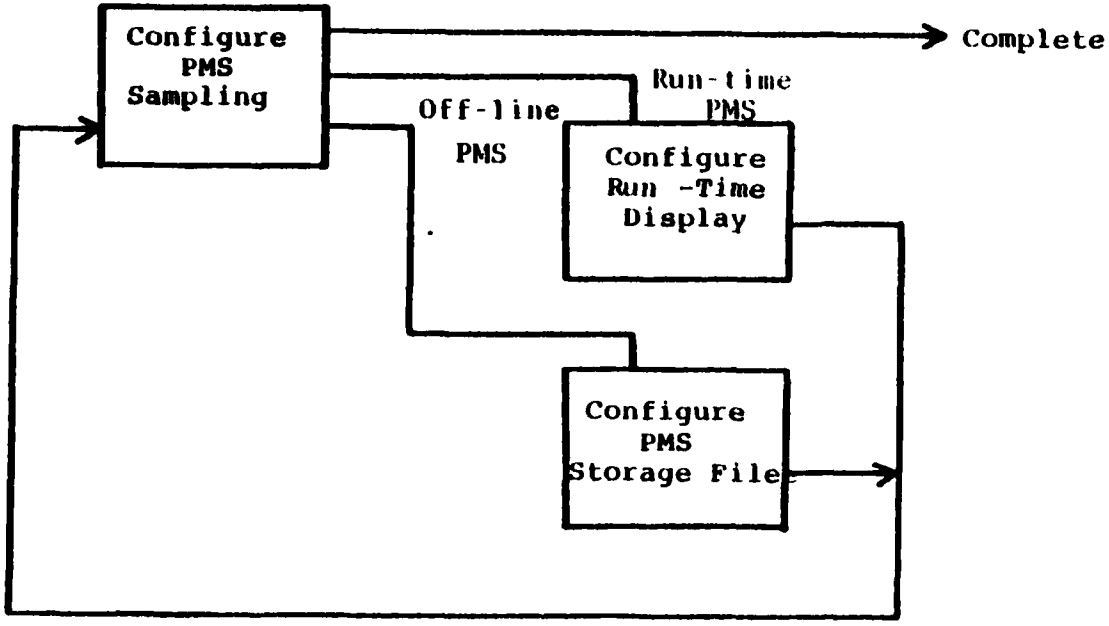
F1C - CONFIGURE PMS

The "Configure PMS" (F1C) phase is the third of four phases of the Run-Time Executive. This phase configures the Performance Monitoring System for the Target System. PMS utilizes the powerful trace capabilities of the RTE DEBUG package to collect information about the TS as the emulation proceeds. Two forms of PMS are available: off-line or run-time.

This phase begins by querying the user for his trace options. These queries are achieved by executing the software defined in "Configure DEBUG" (F1B).

For off-line PMS, an I/O is configured to collect the trace data and store that data in a disk file. This file is saved for later analysis, as defined in Section 19.0,"PMS Off-Line Analysis". The user is permitted to collect PMS data for any, or all, PE's in his Target System.

For run-time PMS, the user is queried for the routines contained in the PMS analysis library; these routines are to be used to produce a display of the selected PE's performance in run -time as the emulation proceeds. If the data from more than one PE is to be displayed, the RTE software in the FCP has to be able to differentiate the data relative to a CS (Computer System). A CS may reside on more than one PE. The library of routines from which the user may select his real-time PMS analysis is identical to that described in Section 19.0.



F1C

Configure PMS

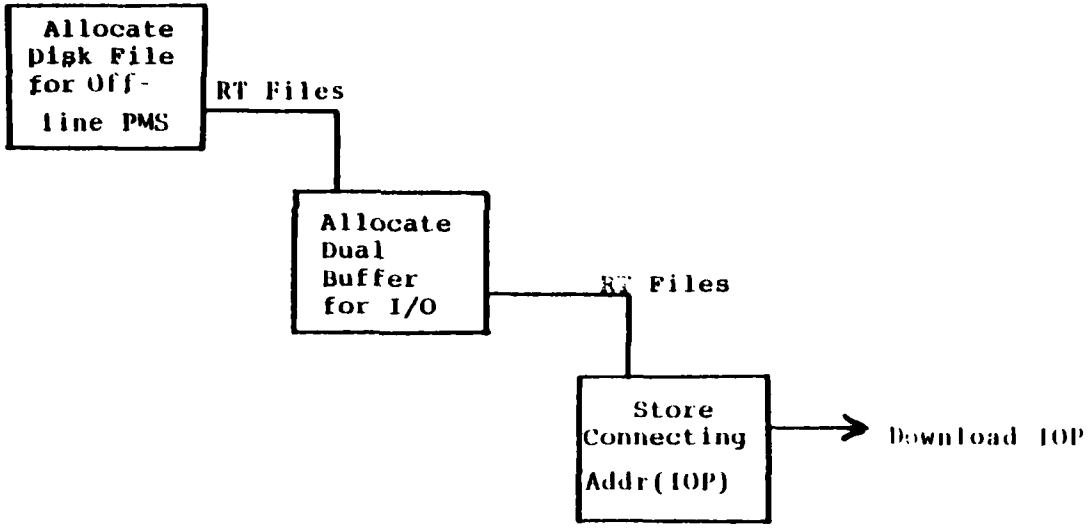
F2C1 - CONFIGURE PMS SAMPLING

This step, "Configure PMS Sampling" (F2C1), is functionally identical to the"Configure DEBUG"(F1B) phase of RTE. Refer to F1B for documentation and charts.

F2C2 - CONFIGURE PMS STORAGE FILE

This step, "Configure PMS Storage File" (F2C2), is executed to complete the configuration of a PE for off-line PMS. The processing of this step is very similar to the processing identified in "Configure Environment" (F1A). This step begins by allocating space on a disk unit attached to the FCP. A dual buffer is allocated and associated with the disk I/O handler. Finally, the address of the dual buffer is down-loaded to the IOP.

During the emulation run, the PE will output PMS trace data which this configured I/O will collect and store on the disk file. This file of data will be saved for later, off-line, analysis.



F2c2

Configure PMS
Storage File

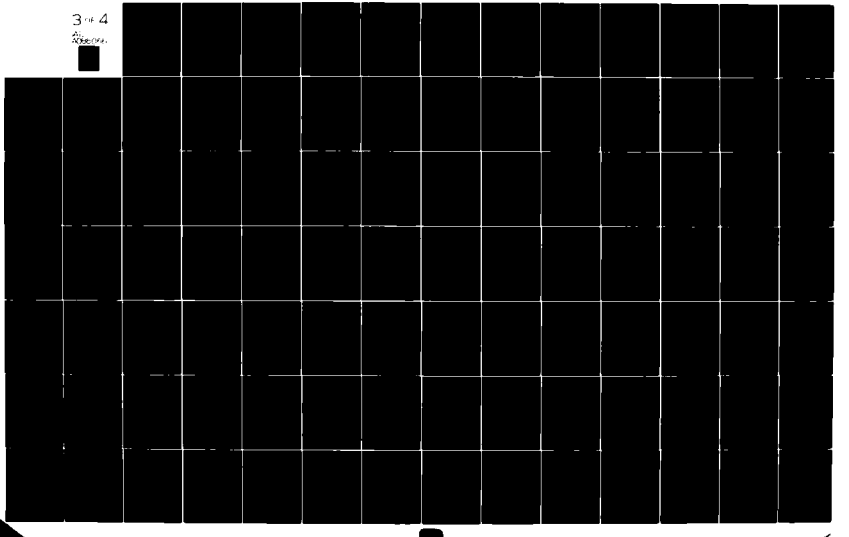
F2C3 - CONFIGURE RUN-TIME DISPLAY

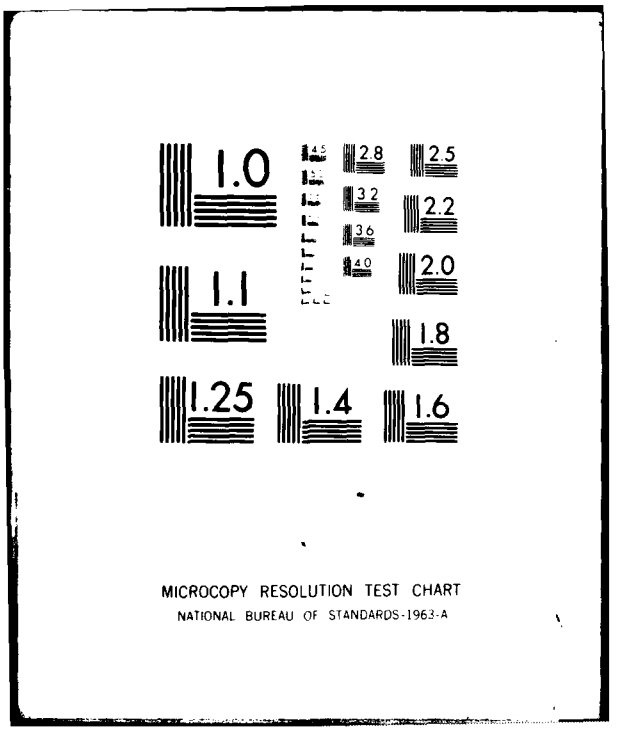
This step, "Configure Run-Time Display" (F2C3), completes the configuration of run -time PMS. This step queries the user for the routines he desires to utilize to construct the run -time display of PMS data. The user may choose routines from a library of analysis routines. These routines are described in greater detail in Section 19.0.

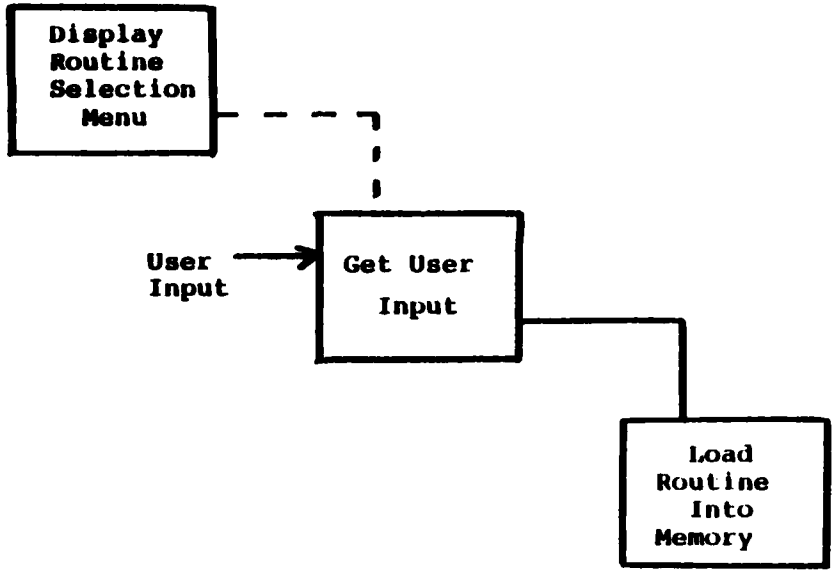
AD-A086 066 HARRIS CORP MELBOURNE FL GOVERNMENT COMMUNICATION SY--ETC F/G 9/2
MULTIPLE MICROPROCESSOR SYSTEM (MMS) DESIGN STUDY. VOLUME II.(U)
MAR 80 F30602-78-C-0114

UNCLASSIFIED RADC-TR-80-33-VOL-2 NL

3 OF 4




1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A



F2C3

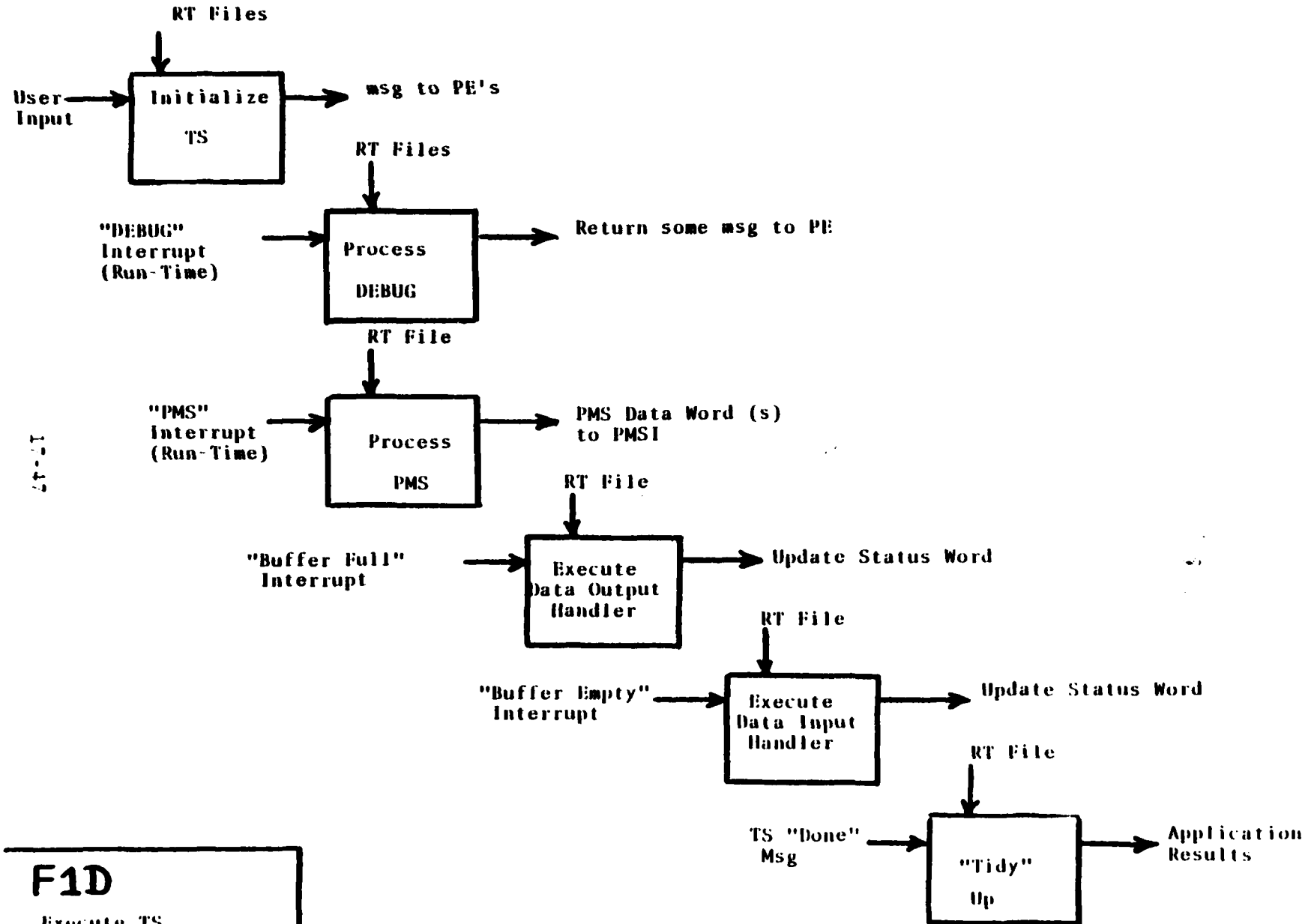
Configure Run Time Display

F1D - EXECUTE TS

This fourth phase (F1D) is called "Execute TS". Once the PE's have been completely configured, the emulation may execute with little or no interference from RTE, which resides in the Facilities Controller Processor. RTE merely provides a set of interrupt handling routines in order to support the emulation processing. These interrupt handlers handle the IOP interface and the PMS interface.

The IOP interrupts the FCP when an I/O buffer has been completely transferred, either to or from a PE. This phase then executes the user configured handler for that I/O to process that buffer for the next I/O.

The PMS interface generates an interrupt to the FCP whenever any PE detects a "run-time" DEBUG/PMS flag set in its MIU or in other internal emulation storage. This phase, in response to this interrupt, displays appropriate information to the user. If the interrupt was caused by a breakpoint, then this phase allows the user to select/deselect his DEBUG/PMS options.

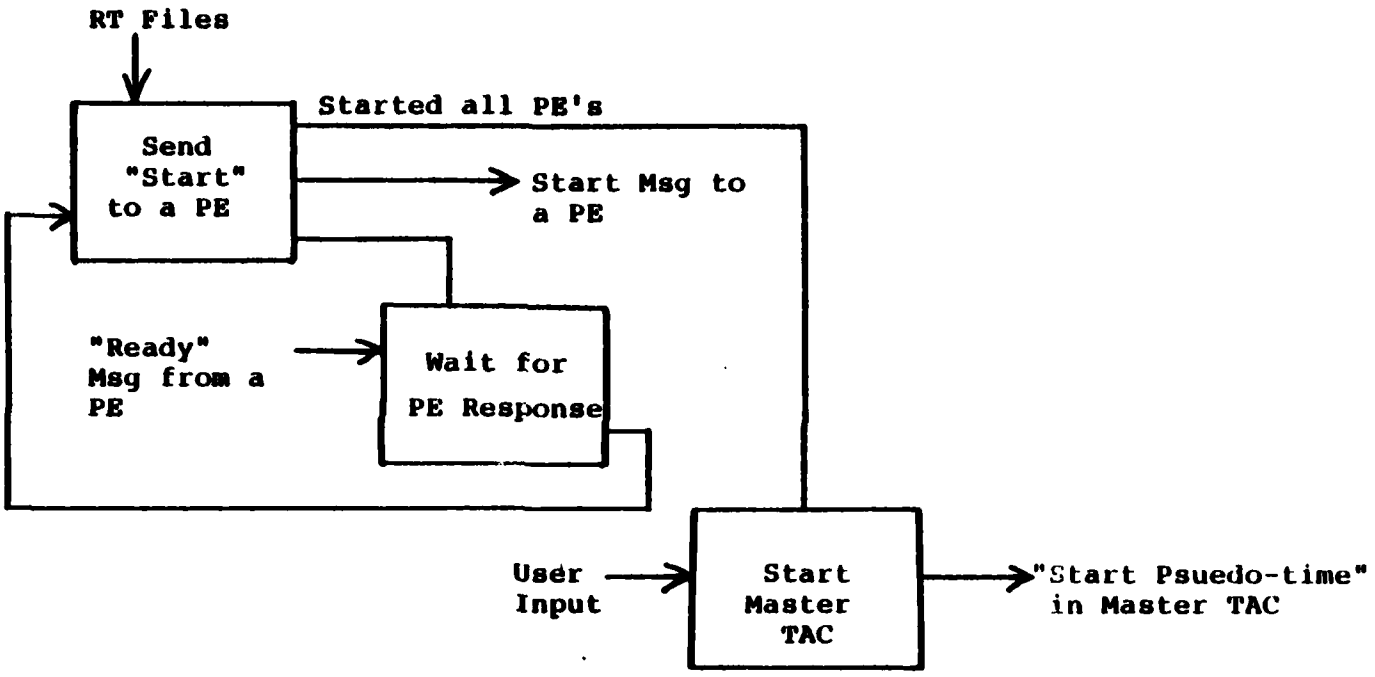


F1D

Execute TS

F2D1 - Initialize Target System

The user's target system has now been completely defined and configured. With this step, "Initialize Target System" (F2D1), the emulation begins. A start message is sent to all PE's in the TS. The user is then queried for a final "GO"/"NO-GO" decision. If the user decides to go ahead with his emulation, the MMS Master TAC is started, and the emulation proceeds.

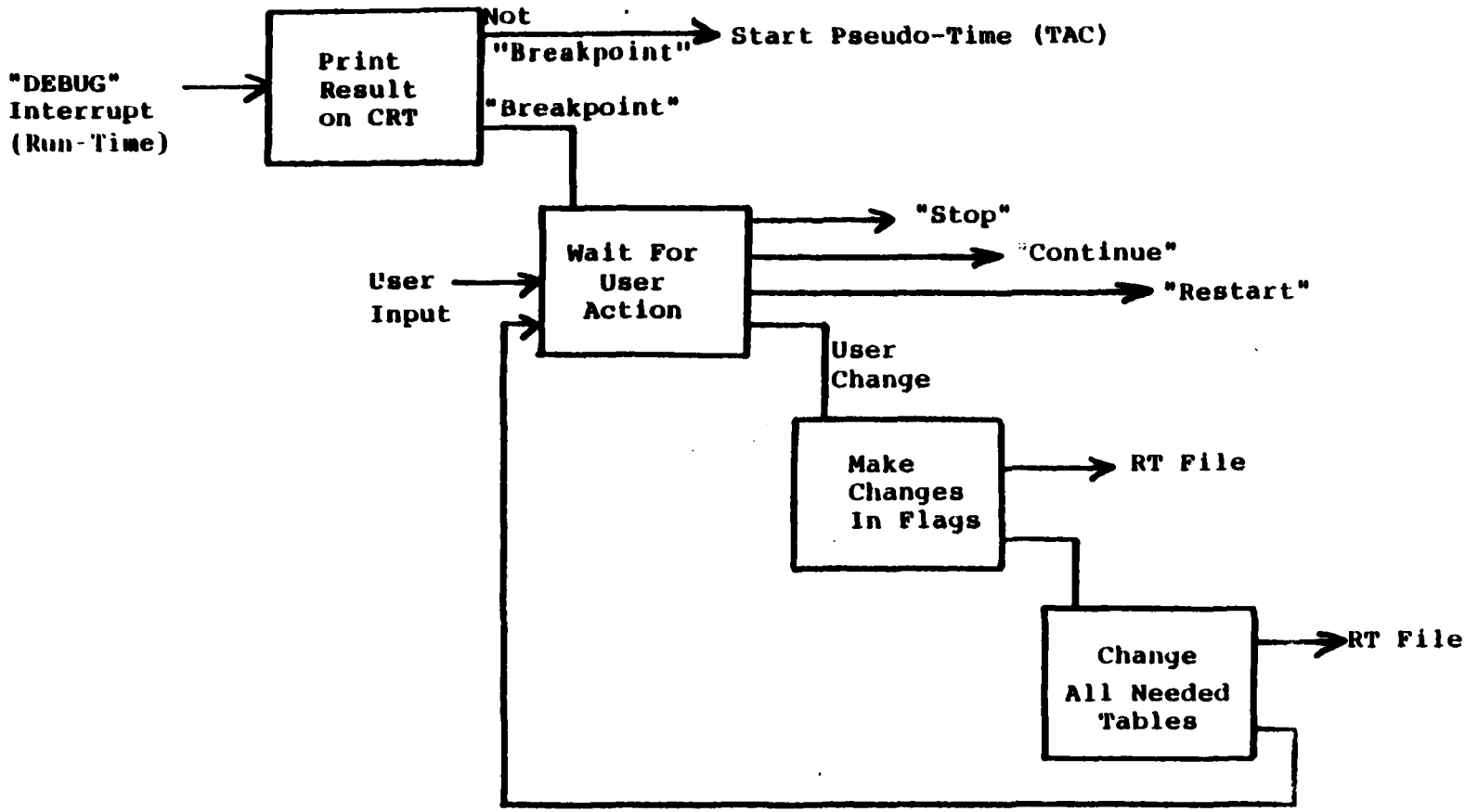


F2D1

Initialize TS

F2D2 - PROCESS DEBUG

This routine, "Process Debug" (F2D2), processes the run -time DEBUG interrupt. The data collected by user-configured option is displayed. If the interrupt was caused by a break-point, then the user is given the opportunity to re-configure his option selections. The user scenario for this re-configuration is identical to that defined for initial configuration. The user may also choose to stop the emulation or restart the emulation from the beginning. Lastly, the user may choose to continue the emulation, either in single step (i.e., single instruction) mode or in free running mode. This continuation restarts the TAC.

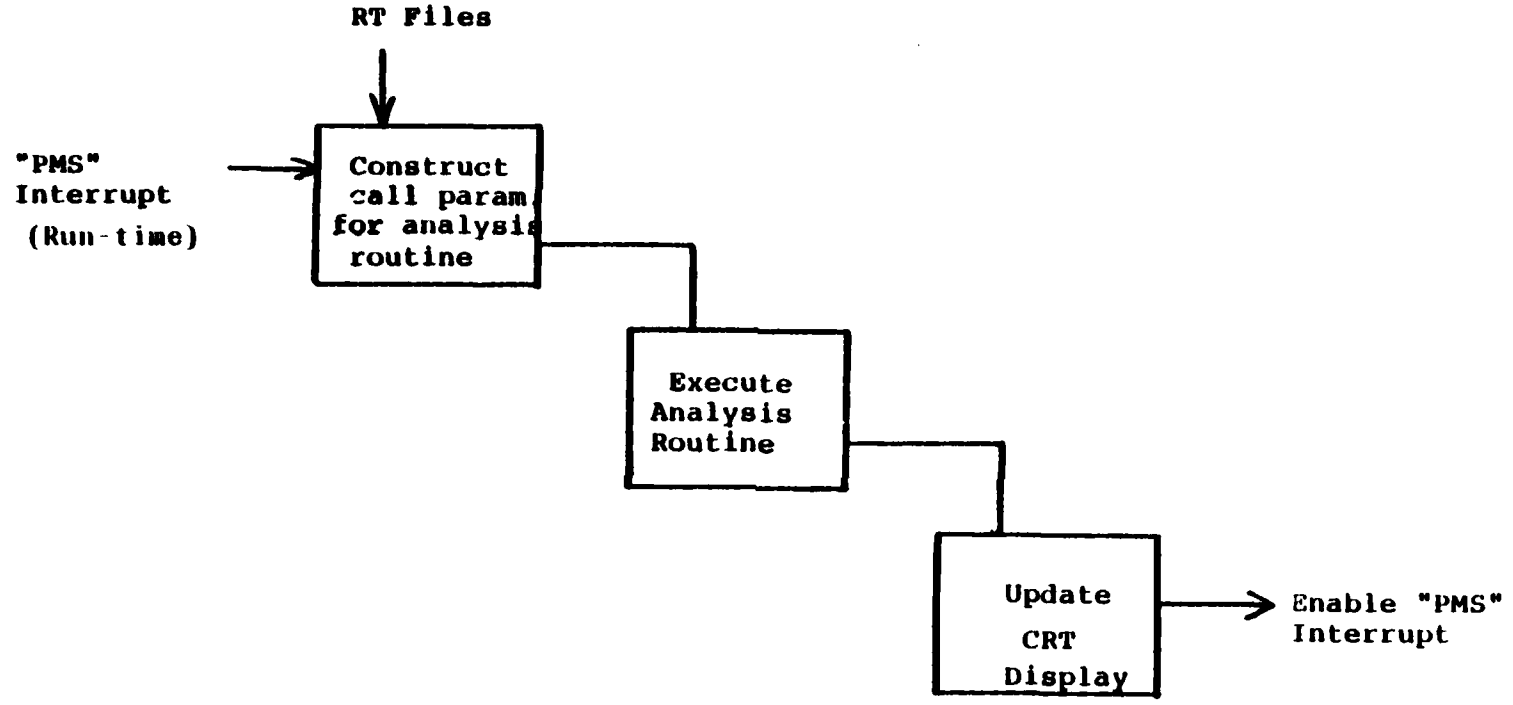


F2D2

Process
DEBUG

F2D3 - PROCESS PMS

This routine, "Process PMS" (F2D3), services the run-time PMS interrupt. This routine reads the PMS trace data from the PE and, in run -time, passes that data to the user-selected PMS analysis routine. Finally, this routine updates the CRT image with the new results of the PMS analysis.



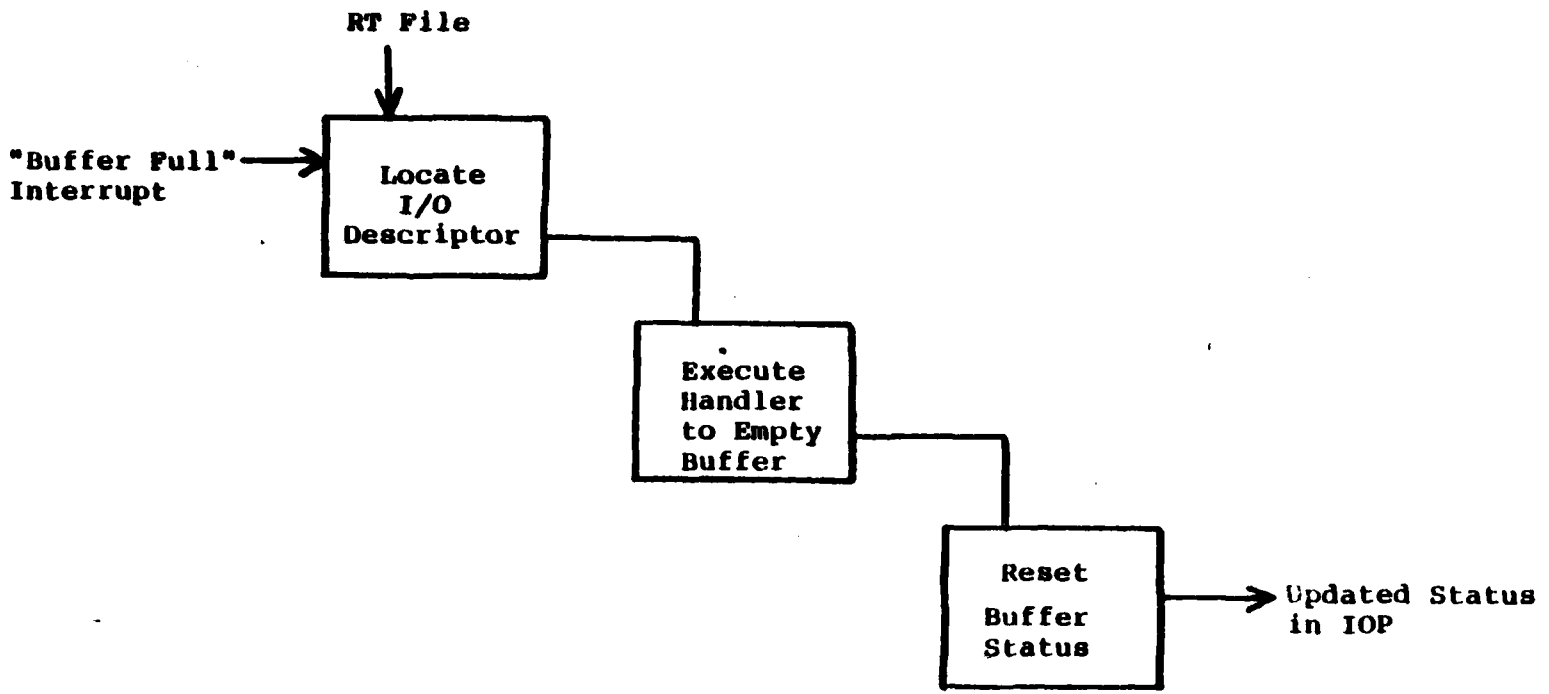
17-53

F2D3

Process PMS

F2D4 - EXECUTE DATA OUTPUT HANDLER

This routine, "Execute Data Output Handler" (F2D4), services the "Buffer Full" interrupt. When a PE produces a data item, it passes that data to the IOP. The IOP contains an I/O descriptor indicating where that data should be stored, either within the FCP or within another PE. This I/O descriptor includes a buffer pointer and current word count. When a buffer is full, the IOP generates an interrupt. This routine services that interrupt. This routine merely executes the previously configured I/O handler. That handler consumes the PE output data (input to the handler) and passes the buffer back to this routine. This routine then updates the new buffer status within the IOP.

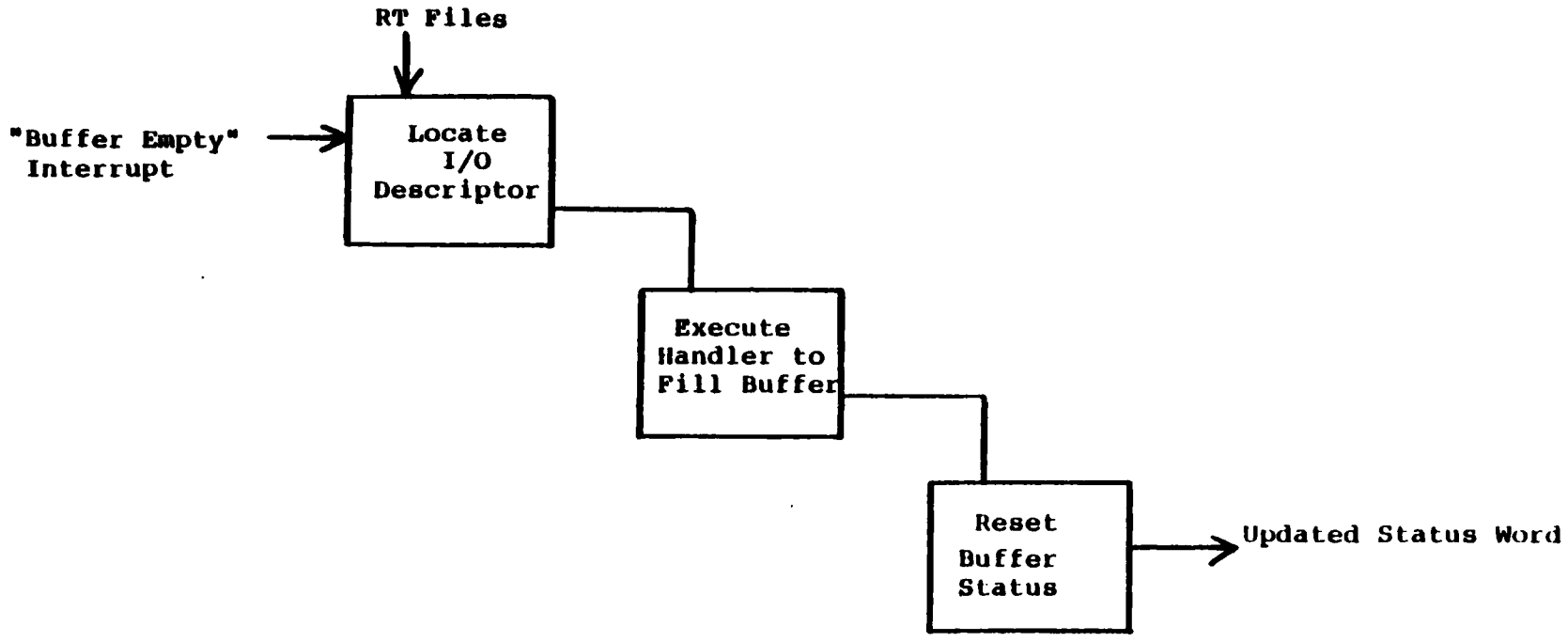


F2D4

Execute Data Output Handler

F2D5 - EXECUTE DATA INPUT HANDLER

This routine, "Execute Data Input Handler" (F2D5), services the "Buffer Empty" Interrupt. When a PE requires an input data item, it issues a request to the IOP. The IOP contains an I/O descriptor indicating the source of input, either the FCP or another PE. This I/O descriptor includes a buffer pointer and a word count. Further, this I/O descriptor also points to another, secondary, buffer. When the initial buffer is empty, the IOP toggles its logic to process the other buffer and generates an interrupt. This routine services that interrupt from the IOP. This routine merely executes the previously configured I/O handler. That handler produces new PE input data (output from the handler) and passes the full buffer back to this routine. This routine completes its processing by updating the new buffer status within the IOP.

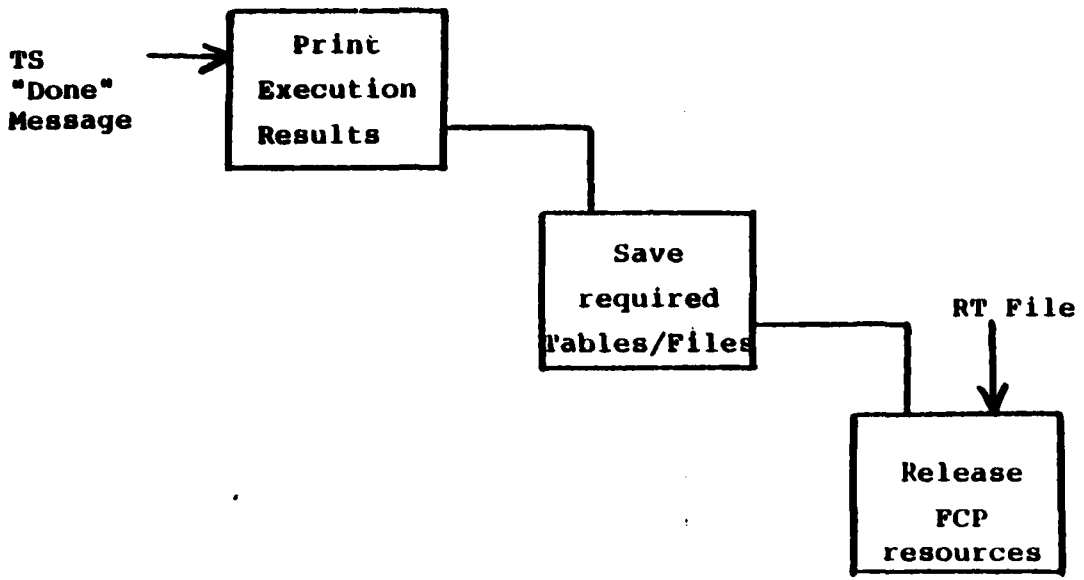


F2D5

Execute Data Input Handler

F2D6 - TIDY UP

This routine, "Tidy Up" (F2D6), completes the processing of the RTE execution phase. This routine saves all pertinent information generated during execution of the emulation. This information includes, as a minimum, any data produced by any PE during the course of the emulation and any off-line DEBUG/PMS data produced. Finally, this routine releases any FCP resources acquired during the emulation run, including buffer memory and temporary disk storage.


TS
"Done"
Message
Print
Execution
Results
Save
required
Tables/Files
RT File
Release
FCP
resources

F2D6

"Tidy Up"

17.1 Abbreviations Used in Section 17

CRT	Cathode Ray Tube
CS	Computer System
FCP	Facility Control Processor
HOL	Higher-Order Language; same as HLL for Higher - Level Language
IEC	Instruction Emulation Code
IEU	Instruction Execution Unit
I/O	Input/Output
IOP	Input/Output Processor
IOPT	Input/Output Port Table
MIU	Memory Interface Unit
MMS	Multiple Microprocessor System
param.	parameters
PE	Processing Element
PEXE	PE Executive
PMS	Performance Monitoring System
PMSI	PMS (Performance Monitoring System) Interface
RT	Run-time Table
RTE	Run-time Executive
SGL	System Generation Loader
TAC	Time Alignment Controller
TS	Target System

18.0 POST MORTEM DUMP

The purpose of the Post Mortem Dump is sc the user can get the contents of any register or storage locations for the TS or for the FCP computer. Just as DEBUG/PMS options allow dynamic dumping of what is happening during execution, the Post Mortem permits the static results of the parts of MMS.

The usual application of PMD would be when the TS has stopped because of an error. The user may choose to stop execution of his TS at any time and then scan the parts of the total system.

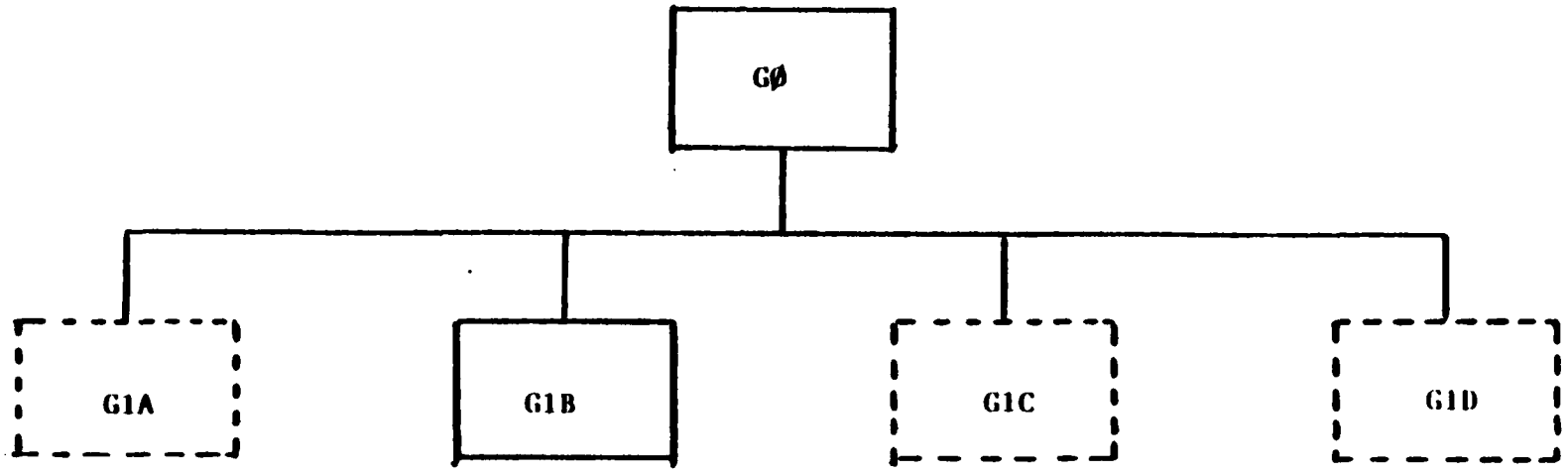
Since the total MMS may be quite large and may thus involve many PE's, the first step should be a listing of the total system which may be accessed by the user. Thus, the user can access any part of the FCP - as the Program Counter, the registers, memory etc. Any module on the Master Segment, called the "Big 7", can also be accessed because a table will exist which contains all bus addresses to these modules. The list will identify all parts of the Computer Systems which are part of the TS, and will also identify which of the CS parts reside on which PE.

Another option available to the user will be that the user can dump to disk every register, all memory values, and other addressable locations from all PE's, from the "Big 7", and from the FCP. This would necessitate much space on the disk. The

user could then print out the total dump, as during the night hours, or the user could use the CRT to selectively scan the dump. This allows another user to quickly utilize MMS.

A suggested refinement is that the results from the TS software be compared against a macrocode table, and that the results be printed as the macro instruction along with the numeric value. This numeric value can be octal, decimal, or hexadecimal. As further refinements are made to the loading of the operating system separate from the user application programs, the user can do dumps relative to program names or to variable names within the named program rather than by address within PE's.

Figure 18.0 illustrates the modules necessary for designing the Post Mortem Dump program. The top level consists of four steps, of which the second level only is more deeply described.



Indicates that this module is not defined at this level.

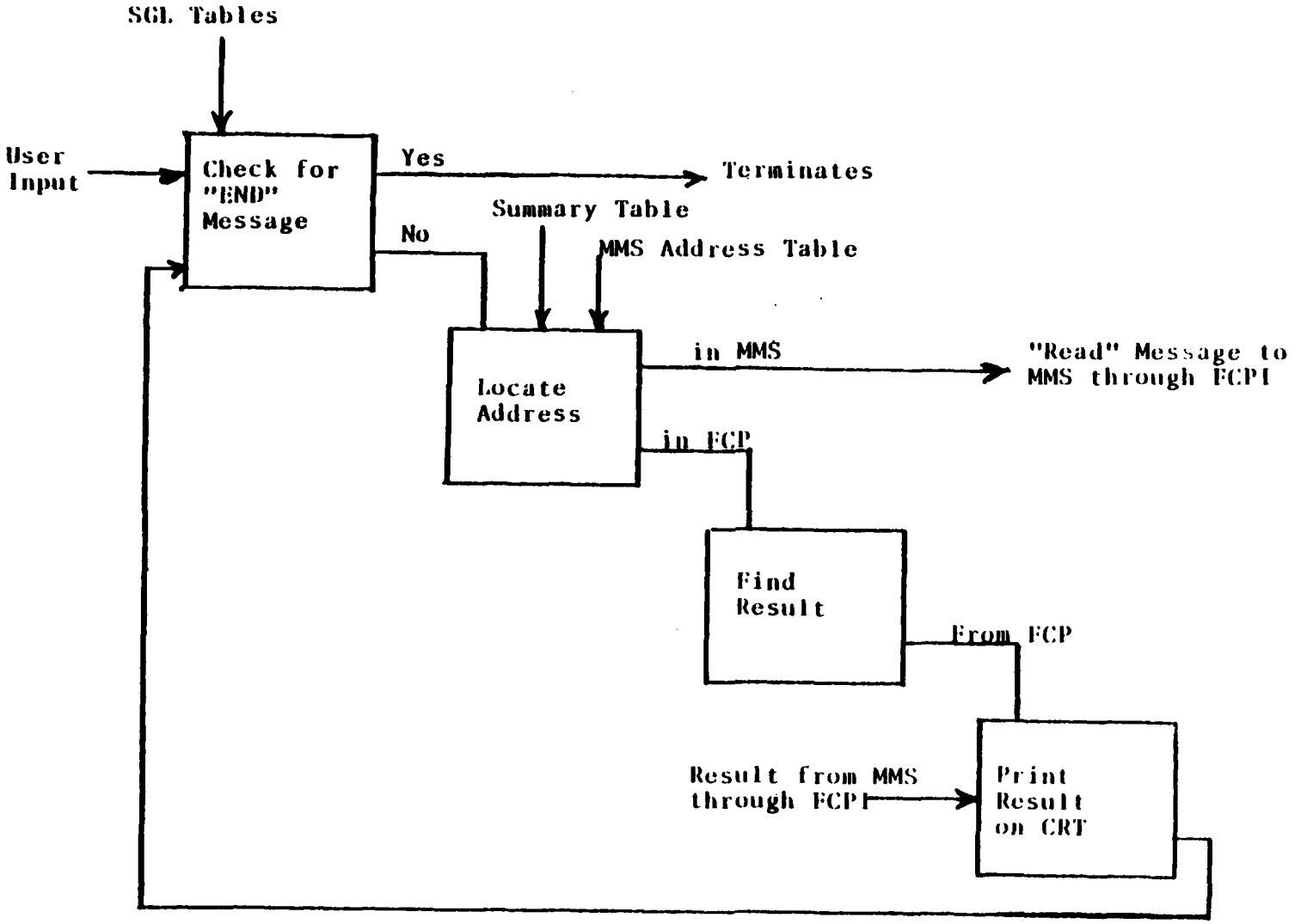
Figure 18.0 Top-Down Design Chart

GØ - POST MORTEM DUMP

The first step is to see what the user wishes to do: to terminate or to check contents of some register or memory location, or to do a total dump.

If the user wants to check contents of some location, register, etc., the Summary Table is listed to show PE numbers, the parts of the TS stored there, the parts of TS stored elsewhere (IOP, SRC, etc.). The user selects FCP or a PE, and then he indicates the address. The result at that location is then displayed on the user's CRT. Then the cycle repeats itself - what does the user now want to do.

If the user wants to do a total dump, all parts of the FCP, the "Big 7", and the utilized PE's are dumped to disk on the FCP.



GØ Post Mortem Dump

G1B - LOCATE ADDRESS

Only this step of "GØ - Post Mortem Dump" needs to be further defined. When the Summary Table is printed, it might be:

1. FCP Computer
2. MBC
3. SRC
4. FCPI
5. PMSI
6. IOP - Bus A, SM
7. S5BC
8. TAC
9. S2CBIU
10. S2BC
11. S2PEØ - PDP-11/45
12. S2PE1 - 8080A
13. S2PE2 - 8080B
14. Dump All

The user would respond with one of the above numbers. Then the options available for that module would be listed from which the user selects. If the user had selected "12", then he might get:

1. BIU
2. LM - TS
3. MI
4. IEU
5. CS - PEXE
6. CS - IEC

7. CS - EBIU

8. CS - SMI

9. IM - PEXE

10. IM - IEC

11. IM - EBIU

12. IM - SMI

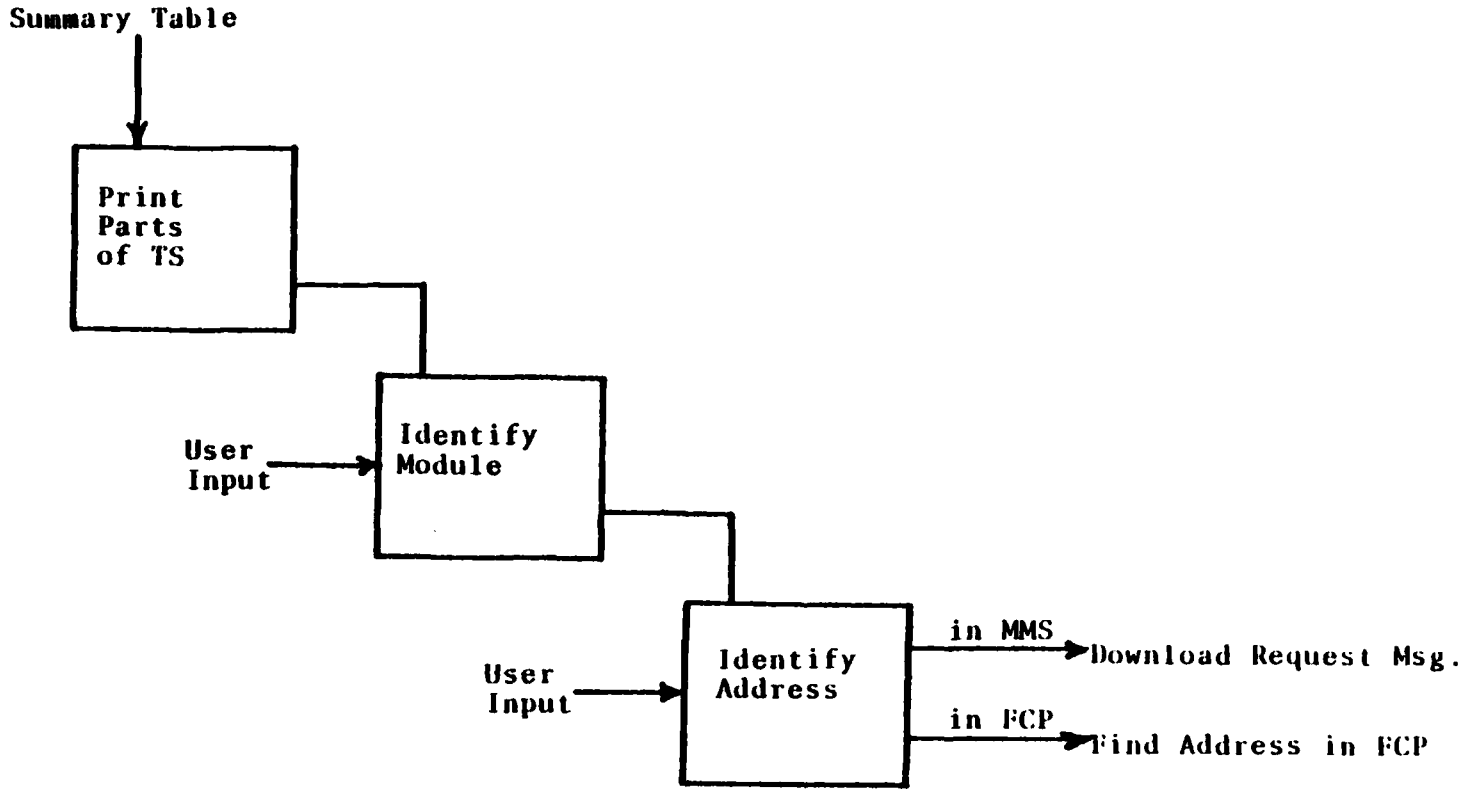
13. PTA

This list identifies those areas in which the user might be interested. If the user wishes to look at memory loaction 4025 of the 8080 TS, he would input:

2 - 4025.

This identifies that the 8080 software of the Target System is stored within Local Memory of PE1, Segment 2.

Any values desired from the MMS PE's have to be requested through FCPI on the Master Bus. These requests are really bus request messages. Any values requested from the FCP are handled through the FCP.



G1B Locate Address

18.1 Abbreviations Used in Section 18

BIU	Bus Interface Unit
CRT	Cathode Ray Tube
CS	Control Store
EBIU	Emulated Bus Interface Unit
FCP	Facility Control Processor
FCPI	Facility Control Processor Interface
IEC	Instruction Emulation Code
IEU	Instruction Execution Unit
IM	Internal Memory
IOP	Input/Output Processor
LM	Local Memory
MBC	Master Broadcast Controller
MI	Memory Interface Unit
MMS	Multiple Microprocessor System
Msg.	Message
PMD	Post Mortem Dump
PE	Processing Element
PE1	PE1
PEXE	PE Executive
PMS	Performance Monitoring System
PMSI	PMS (Performance Monitoring System) Interface
PTA	Pseudo-time Accumulator
SGL	System Generation Loader
SM	Shared Memory
SMI	Shared Memory Interface
SRC	Shared Resource Controller
S2BC	Segment 2 Bus Controller
S2CBIU	Segment 2 Control and Bus Interface Unit

S2PEØ	Segment 2 PEØ
S2PE1	Segment 2 PE1
S2PE2	Segment 2 PE2
S5BC	Segment 5 Bus Controller
TAC	Time Alignment Controller
TS	Target System

19.0 PMS OFF-LINE ANALYSIS

Whenever the user has gathered a lot of data relative to conditions or events that resulted during the execution of a program, he will amass this data on the disk for later evaluation.

The PMS Off-Line Analysis Program will allow the user to easily process this large amount of run data. The user will be able to list the selected data, to develop a frequency table, to do straight line and curve fits, and to do statistical analysis on the data in order to derive a numeric measure.

This package will be developed as a main program, and a library of routines selectable from the main program. The user can enhance this library with new routines. Note that this approach of a main routine accessing a library of routines allows the library to be utilized during run-time. (See explanation under "Run-Time Executive", Section 17) The output can be either lists, tables, or graphs; the output can be produced on either the line printer or on the CRT. The graphs for the line printer will not be as smooth as those produced on the CRT.

The routines available in the library will include:

- Display of Data
  - Table printout
  - Scattergram or Scatter Diagram
  - Histogram
  - Table printouts from Statistical Analysis Routines
  - Line fits from fitting data to equations

- Statistical Analysis
  - Frequency Analysis
  - Central Tendency
  - Measure of Variation
- Line Fits
  - Broken Line
  - Straight Line by Selected Points
  - Straight Line by Least Squares
  - Second Degree Parabola by Least Squares

Each of these routines will be further detailed before the design of the total PMS Off-Line Analysis Program is discussed.

19.1 Display of Data

The data can be organized and displayed as either printouts or listings, or can be displayed graphically on some coordinates as a graph.

The plot routines should allow the user to select the mimimum and the maximum values for the plots. The routines can then be written to fit one of two ways. The first is that the data will be plotted on the resulting scale, even if the incremental steps are not numerically logical. The second way would be that the plot would closely approximate the min-max range, but that the maximum value would be extended to obtain numerically logical increments. Thus, if the minimum is "0" and the maximum is "92", and since the horizontal axis is 100 units long, the maximum value would be extended to "100".

19.1.1 Table

Theory: All PMS-collected data has been stored on disk. To print the results in a tabular form, the user should be able to print all collected data or to print selected data from the total. Because the amount of data per PMS event will vary, this specialized routine should be able to know column headings relative to the PMS event. This routine should be callable for printouts for the line printer, for the CRT, or for any other device - with the output form being specified by a number. The final criteria should be that this routine be callable by the other routines to handle their output.

Input: KEY, which specifies the PMS event code.
FILE of PMS data
TYPE, which specifies either line printer, CRT, or an additional output peripheral.

Output: The table, with headings, printed on the line printer, CRT, or other peripheral.

19.1.2 Scattergram or Scatter Diagram

Theory: The data to be utilized may be either the original data or it may be the derived frequency table. The X-values will be scaled along the horizontal axis, and the Y-values will be scaled along the vertical axis. Each pair of X - Y values will represent a point within the graph. The accuracy will be dependent upon the scales selected. It is possible that many points will coincide when depicted on the line printer but will appear more dispersed when depicted on the CRT.

Thus, the purpose of a scattergram is to show relative relationships of the data points.

Input: File of PMS data (X - Y values).

LENGTH, which specifies number of X - Y values.

TYPE, which specifies either line printer, CRT, or an additional output peripheral.

Output: The plot on desired output.

19.1.3 Histogram

Theory: Once the user has grouped the results into fixed intervals, it is easier to comprehend the frequencies of events occurring in all intervals through a pictorial representation. A histogram reflects the intervals on the abscissa (X-axis) and the frequencies on the orignate (Y-axis). Thus, a histogram is a bar graph, with the height reflecting the frequency which occurs within the range.

Input: File of PMS data (X - Y values).

LENGTH, which specifies number of X - Y values.

TYPE, which specifies either line printer, CRT, or an additional output peripheral.

Output: The plot on desired output. The bars can be one character wide, or can be several characters wide. A good approach would be that of flexible width, relative to the graph; the center would be blank, i.e., only the outline of the box would be printed.

19.1.4 Table Printouts from Statistical Analysis Routines

Theory: As the different statistical analysis routines are applied to the data, several different tables are produced.

The descriptions and nature of these tables are discussed as part of those routines. This description is included here to indicate another type of table printouts.

19.1.5 Line Fits From Fitting Data to Equations

Theory: As the different line fit routines are applied to the data, several curves can be produced. The descriptions of these curves are discussed within the line fit routines. This description is included here to indicate another type of printouts.

19.2 Statistical Analysis

As a beginning on analysis of the data or analysis of the frequency table, the following types of routines are included: an analysis of the frequency table to numerically show the relationships of the resulting intervals; numeric measurements of the cohesiveness of the data; numeric measurements of the dispersion of the data.

In all cases, the input will consist of the table of X-values and Y-values and will include the variable LENGTH to specify the length of this table. Another table of length n, where n reflects the total tests available in the routine, will consist of flags correlating to tests. Thus, a user can do all calculations or can select those he desires.

The output from these routines can be a table of results, a printout on the CRT, or a line printer output. This selection is by number, and is a user input selection.

19.2.1 Frequency Analysis

Theory: Once the data has been grouped into fixed intervals, the user may desire to assess the table of intervals. A table printout can present the analysis of the frequencies.

Method:

First column is the range of the interval.

Second column is the midpoint value of the interval.

Next column is interval frequency, or number of occurrences in this interval.

Next column is relative frequency, or the interval frequency divided by total occurrences.

Next column is frequency percent, or the relative frequency expressed as a percent.

Next column is cumulative of the relative frequency, which is a summation of the relative frequencies.

Last column is cumulative of the frequency percentage, or summation.

19.2.2 Central Tendency

Theory: The user may desire some measure of his original data or of his grouped data which resulted when he did a frequency table. Several descriptive measures exist. This routine has several measures of the cohesiveness of the data.

Method:

1. Mean or Arithmetic Average of the data.

$$\bar{x} = \frac{1}{n} \sum_{i=1}^{n} X_i$$

2. Median or the middle observation when the data are arranged in order of magnitude.

3. Mode or observation of maximum frequency, which refers to the midpoint of the class interval with maximum frequency.

19.2.3 Measure of Variation

Theory: Once the user has grouped his data into fixed intervals, he may wish to summarize that data by some measures of variation (or dispersion). These measures are grouped together to allow the user to select only those measures which he requires.

Method:

1. Range of the data, which is the largest minus the smallest.

$$\text{Range} = X_{max} - X_{min} = X(N) - X(1)$$

2. Average Deviation - the average amount by which the items differ from the mean.

$$AD = \frac{\Sigma(|X|)}{N} = \frac{\Sigma(|X - \bar{X}|)}{N}$$

3. Standard Deviation (or Root-Mean-Square Deviation) - measures the deviations from the mean.

$$\sigma = \sqrt{\frac{\Sigma(X^2)}{N} - \bar{X}^2}$$

4. Variance or Sample Variance - the square of the standard deviation of any distribution.

$$S^2 = \frac{\sum_{i=1}^{N} (X_i - \bar{X})^2}{N-1}$$

19.3 Line Fits

The user can apply any straight line or curve fitting techniques to his data. Any of the resulting lines will represent his data. Thus, a family of straight lines can result; a family of parabolas can result; families of higher-order curves can result. Thus, the user cannot select a line which does not represent his data. But, a line of each family type best fits the user's data, and there exits one line and only one line which best fits the data. The user has to decide this for himself.

The methods presented here are those which are most popular and which have greatest reliability. The needed input consists of the X - Y values and length of the table. When the user selects an output medium (as line printer, CRT, or other), this will constitute his output.

19.3.1 Broken Line

Theory: Assuming that the user has the data in the form desired (as a frequency table), he may wish to graphically depict the data in a simple form. A broken line graph plots the data with connecting lines to reflect behavior of the data.

Method: The output on the CRT will reflect a truer picture of this plot because it can better draw the connecting lines. The points should be indicated on all output, with an attempt to indicate the connecting lines. The size of the plot, and thus the scaling, will follow that outlined in paragraph 19.1.

19.3.2 Straight Line by Selected Points

Theory: When fitting the data to an observation line, the points are selected to be those near the two ends of the frequency distribution. Therefore, the two points representing the average of the first and last intervals become the end points of this straight line. The output will be more meaningful if a scattergram is also printed with this graph.

Method: Solve the pair of equations:

$$Y_1 = a + bX_1$$

$$Y_2 = a + bX_2$$

to get a and b. Then the resulting line will be along the line: $Y_n = a + bX_n$.

19.3.3 Straight Line by Least Squares

Theory: When fitting the data to a linear function, the observations will deviate from the assumed functional relationship. It is assumed that the values of X do not have zero errors, but that these errors are small in comparison to the variability of Y. Therefore, this method produces the one line from which the sum of the squares of the errors is the smallest possible for any line fitted to the data.

Method: Need to solve for a and b in the following two normal equations:

$$Na + b\Sigma X = \Sigma Y$$

$$a\Sigma X + b\Sigma X^2 = \Sigma XY$$

where N is number of observations

X is the input X-values

Y is the input Y-values.

With the values of a and b, we get the line $Y = a + bX$.

19.3.4 Second Degree Parabola by Least Squares

Theory: The method of least squares can also be applied to the data to result in a second-degree parabola which is more likely to be right than any other second-degree parabola if the residuals are normally distributed. In order to obtain the equation of a parabola, $Y = a + bX + cX^2$, we need to solve three equations.

Method: Need to solve for a, b, and c in the following set of three normal equations:

$$Na + b\Sigma X + c\Sigma X^2 = \Sigma Y$$

$$a\Sigma X + b\Sigma X^2 + c\Sigma X^3 = \Sigma(XY)$$

$$a\Sigma X^2 + b\Sigma X^3 + c\Sigma X^4 = \Sigma(X^2Y)$$

With the values of a, b, and c, we have the equation of the curve: $Y = a + bX + cX^2$.

19.4 General Routines

A library of general routines should be available to allow the user to scan the file of PMS data. The output could be a total of the number of "hits" or occurrences of a specified PMS event, or key. Or, this output could be the file created for some of the statistical analysis programs or for the plots.

When the user wishes to reduce the data file to a frequency table, which reflects the total number of occurrences within some range of X-values, this routine has to be available. This routine allows the user the option of applying the original data set or a frequency table to the other routines within this PMS library of routines.

19.4.1 Summary

Theory: The user can use any file of data that exists within a file. By specifying the total number of colums per set of data, and then identifying one of the columns as the "key", the user can then collect all rows or occurrences in which a specified condition happens within the named column.

Method: The user identifies total columns per table, or file. Then, by number, he identifies the important column. Then, he specifies the condition of that column in which he is interested. The output could be a numeric value showing length of the scan, and a file of that length which contains all "hits".

19.4.2 Frequency

Theory: The user should be able to apply a set of range values to the set of X-values, and to develop a set of Y-values which indicates the number of occurrences within each of the increments.

Method: The user specifies the minimum and maximum values of the X-scale. He then decides the step values to apply to this X-scale. The input table is scanned to count total events within each incremental block. This frequency table is utilized as input to many of the other routines, but the results of these other routines can vary because of the choices of the user within this routine.

19.5 References

"Engineering Statistics", Albert H. Bowker and Gerald J. Lieberman, Prentice-Hall, 1972.

"Introduction to Statistical Data Processing", Theodor D. Sterling and Seymour V. Pollack, Prentice-Hall, 1968.

"Elements of Statistical Method", Albert E. Waugh, McGraw-Hill Book Co., 1952.

"Statistical Methods", Herbert Arkin and Raymond R. Colton, Barnes & Noble.

19.6 Top-Down Design

The PMS Off-line Analysis program consists of the main driver routine which can call any of utility packages as sub-routines. The library of utility programs can be expanded by the user. Figure 19.6 reflects the charts included in this design. Not much detail is required in the functional design of this package.

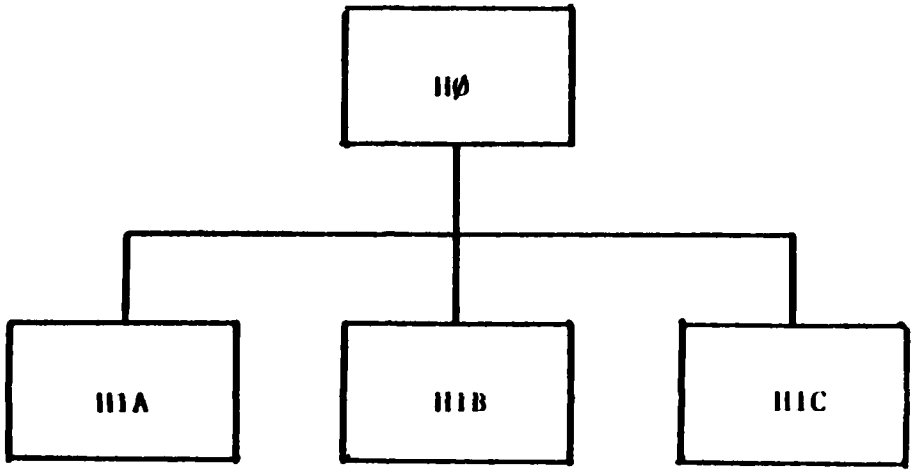
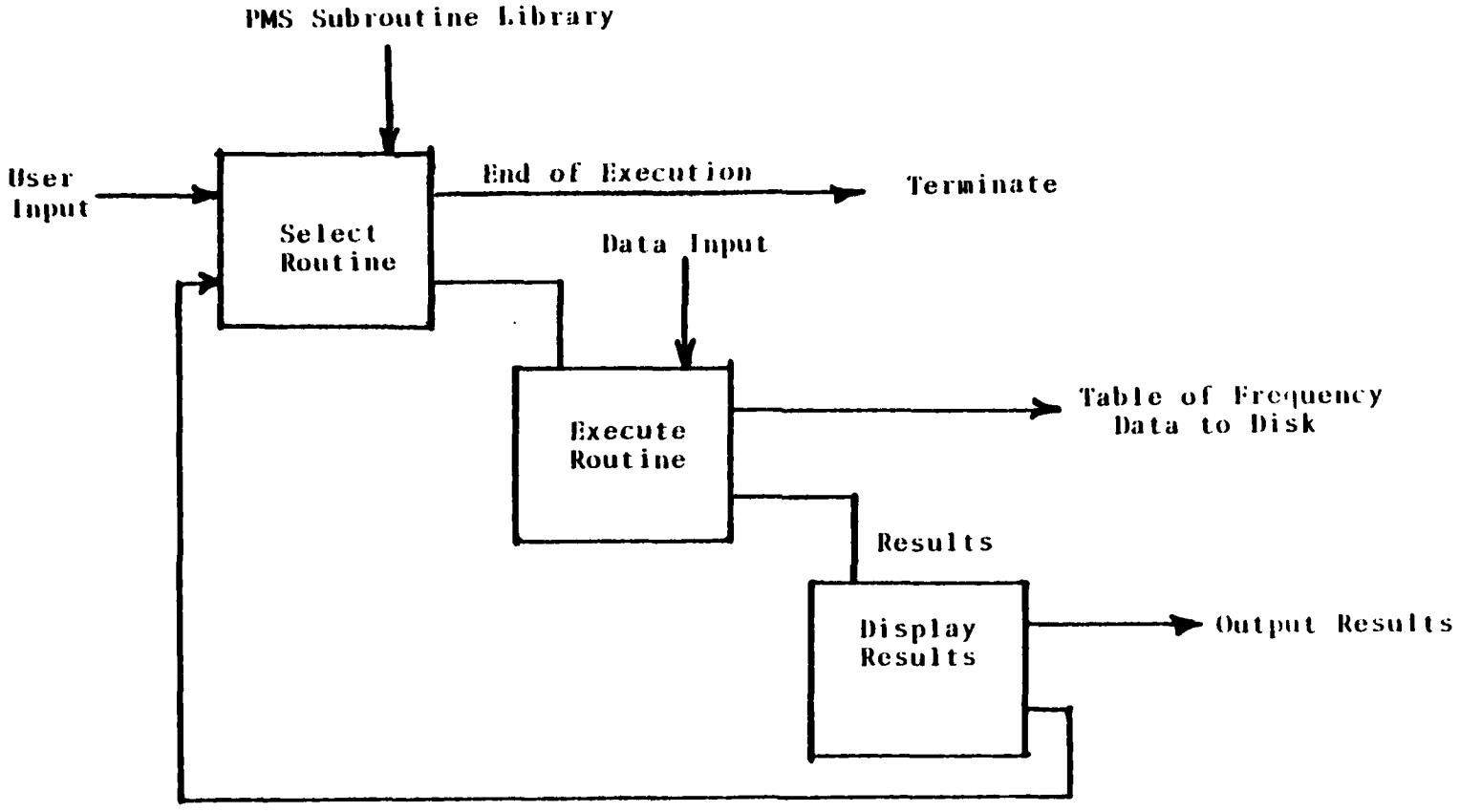


Figure 19.6 Top-Down Design Chart

HØ - PMS OFF-LINE ANALYSIS

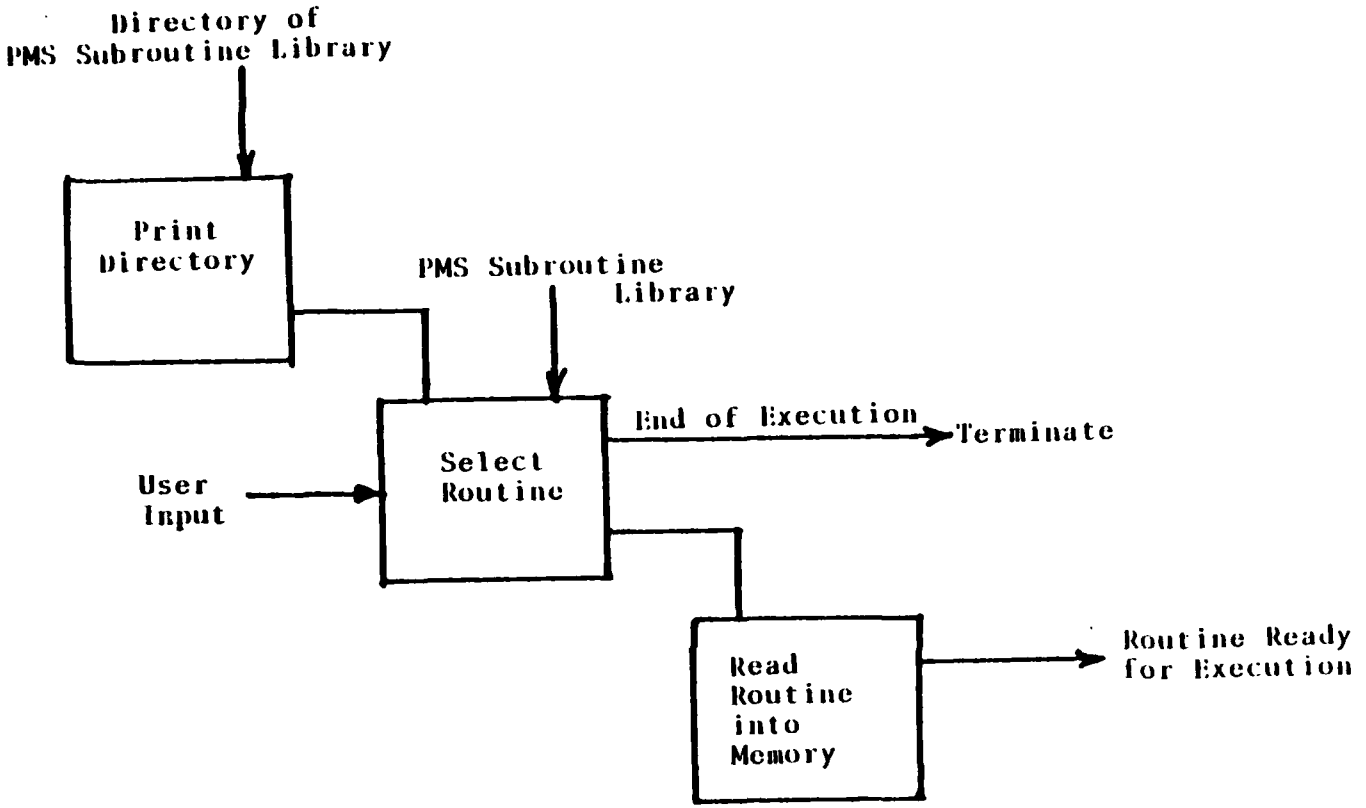
When the PMS Off-Line Analysis package is called, the user sees the list of all routines available to him. He selects one-by-one those routines that he wants processed. If the user doesn't want to run in the interactive mode, he should be able to list, at one time, all routines to be processed.



HØ PMS Off-line Analysis

H1A - SELECT ROUTINE

When the user initiates the PMS Off-Line Analysis program or when he prints the results of a previously-run routine, he can select the routine from a list of those available. Once one is selected, that routine is called; control passes to that routine.

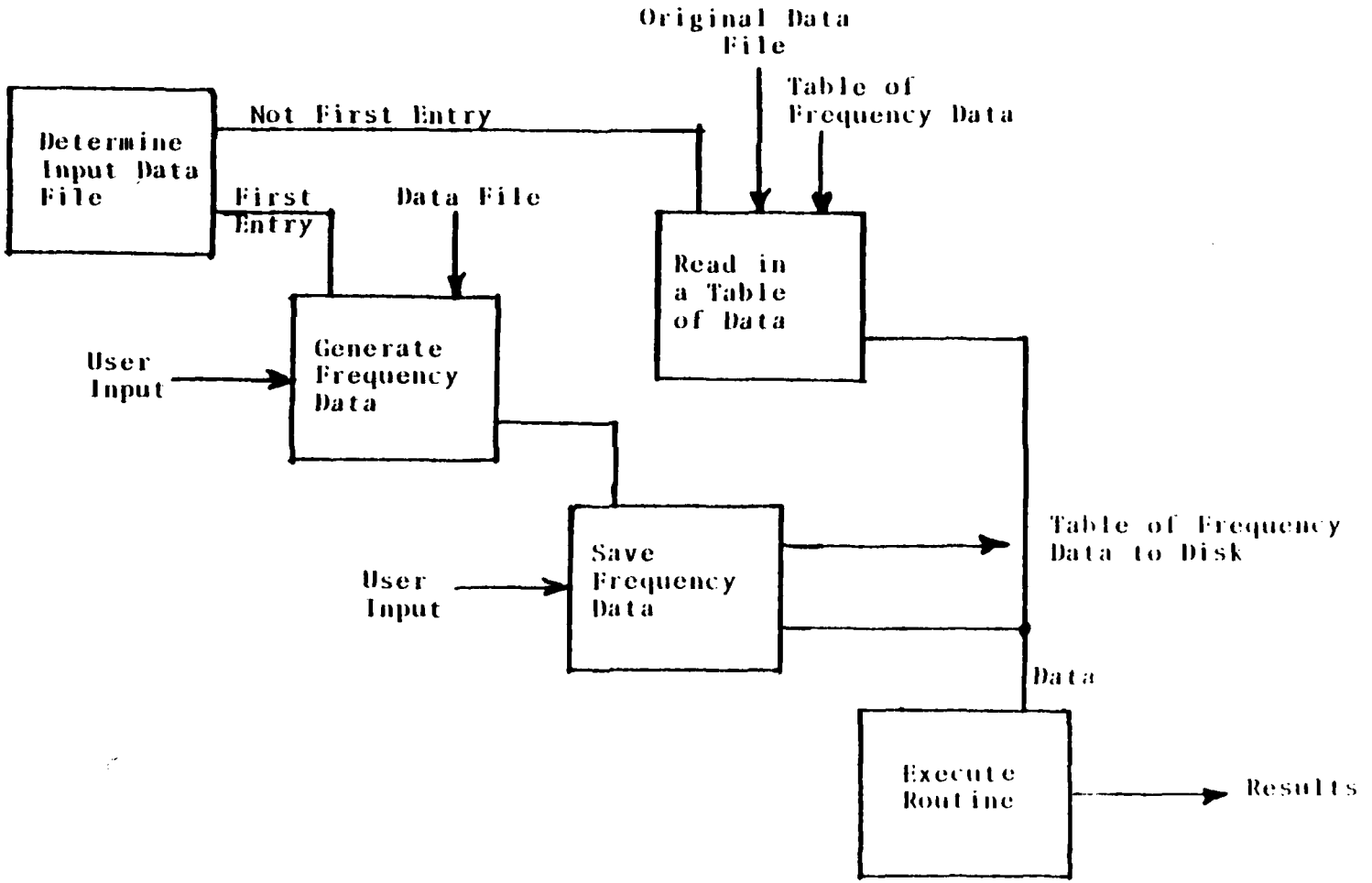


H1A Select Routine

H1B - EXECUTE ROUTINE

If the user has a frequency table ready to access, this file is read into memory. If the user wishes to generate that data file, he processes the original data file through the routine that creates his frequency table values. Once this table of frequency values is created, the user may wish to save this.

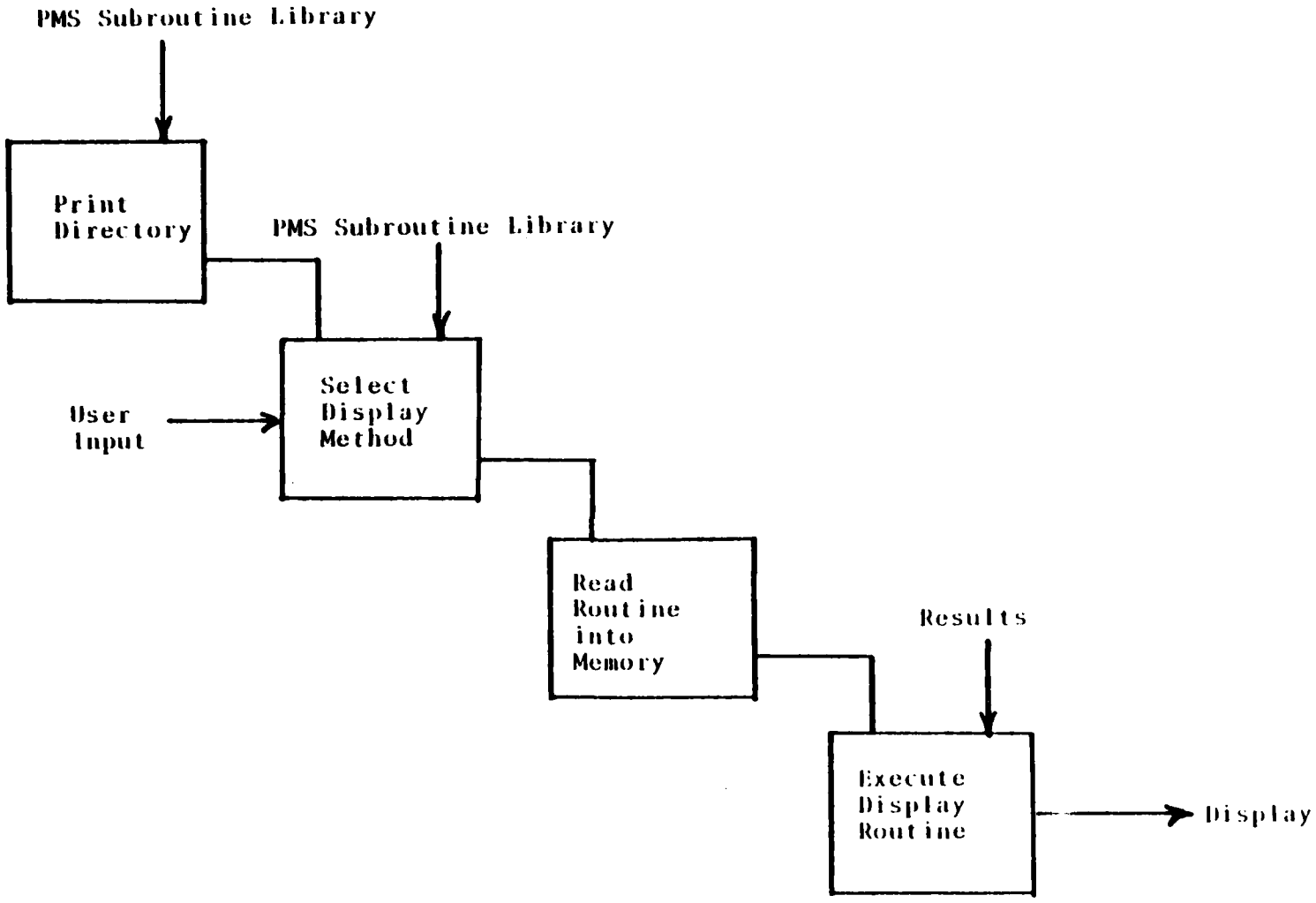
The desired routine is then executed. The input may be the newly created frequency table, a previously-created frequency table, or the original data file. The output will be a table of values which forms the input into a display routine.



H1B Execute Routine

H1C - DISPLAY RESULTS

Most routines have the ability to automatically display that data. The user can utilize the display as part of that routine or he can save the results to use as input to another one of the routines.



H1C Display Results

19.7 Abbreviations Used in Section 19

CRT	Cathode Ray Tube; a terminal
PMS	Performance Monitoring System

20.0 TARGET SYSTEM

Target System (TS), in the context of MMS, describes the system to be emulated. A TS emulation is included as part of this MMS package for two reasons. First , it is not possible to show that MMS, once built, works or that it works correctly without having a test system. Second, the TS emulation package illustrates how to utilize MMS and how to emulate the different techniques.

Section 20.1 discusses the format of all emulation modules. Section 20.2 identifies a proposed TS. Section 20.3 discusses the emulation of PDP-11/45, which is part of the proposed TS, while Section 20.4 then discusses the rest of the proposed TS - the INTEL 8080. Section 20.5 is included to discuss the approach towards the handling of registers of the emulated computers.

20.1 Emulation Modules

Section 14 covered the "Microcode Assembler" and how to organize each emulation module. Several header blocks were created, in order that the user could "fine tune" some of the emulators; this is because the user has these same choices when installing the actual hardware.

Section 16 covered "System Generation Loader (SGL)", and how SGL processed these emulation modules.

This section will discuss the steps for creating the emulation modules for the proposed TS. Figure 20.1 identifies the family chart for the Target System emulation.

Thus, every Computer System will follow this format to create the module for the instruction set and for the separate modules to emulate the individual parts of that Computer System. Also, suppose the emulated Computer System involves a disk controller with two disk units. If the user only wishes to access one disk unit, then he can emulate the necessary portions to allow I/O and need not emulate the controller.

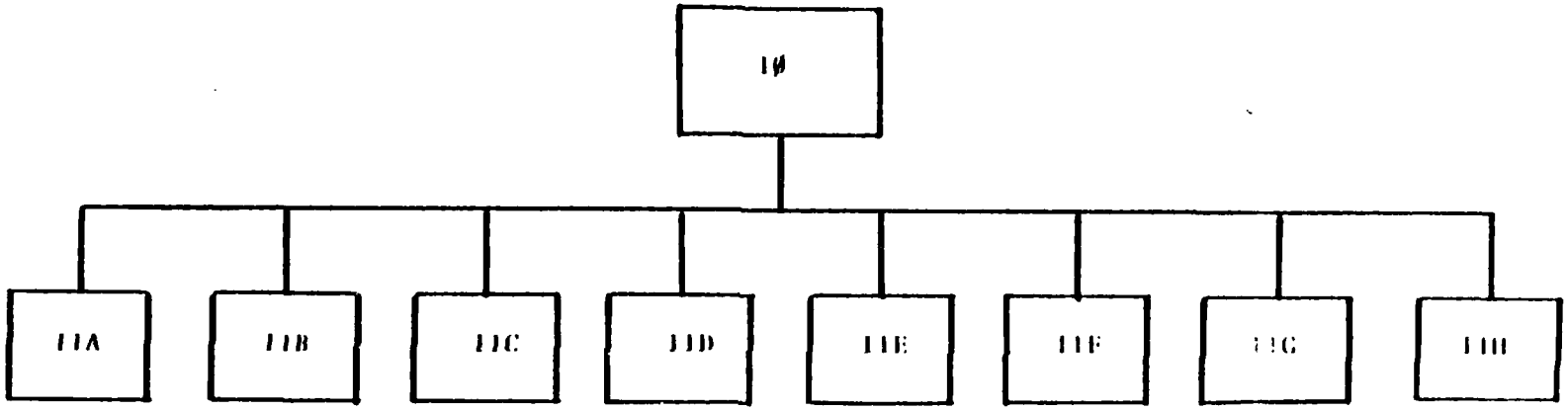


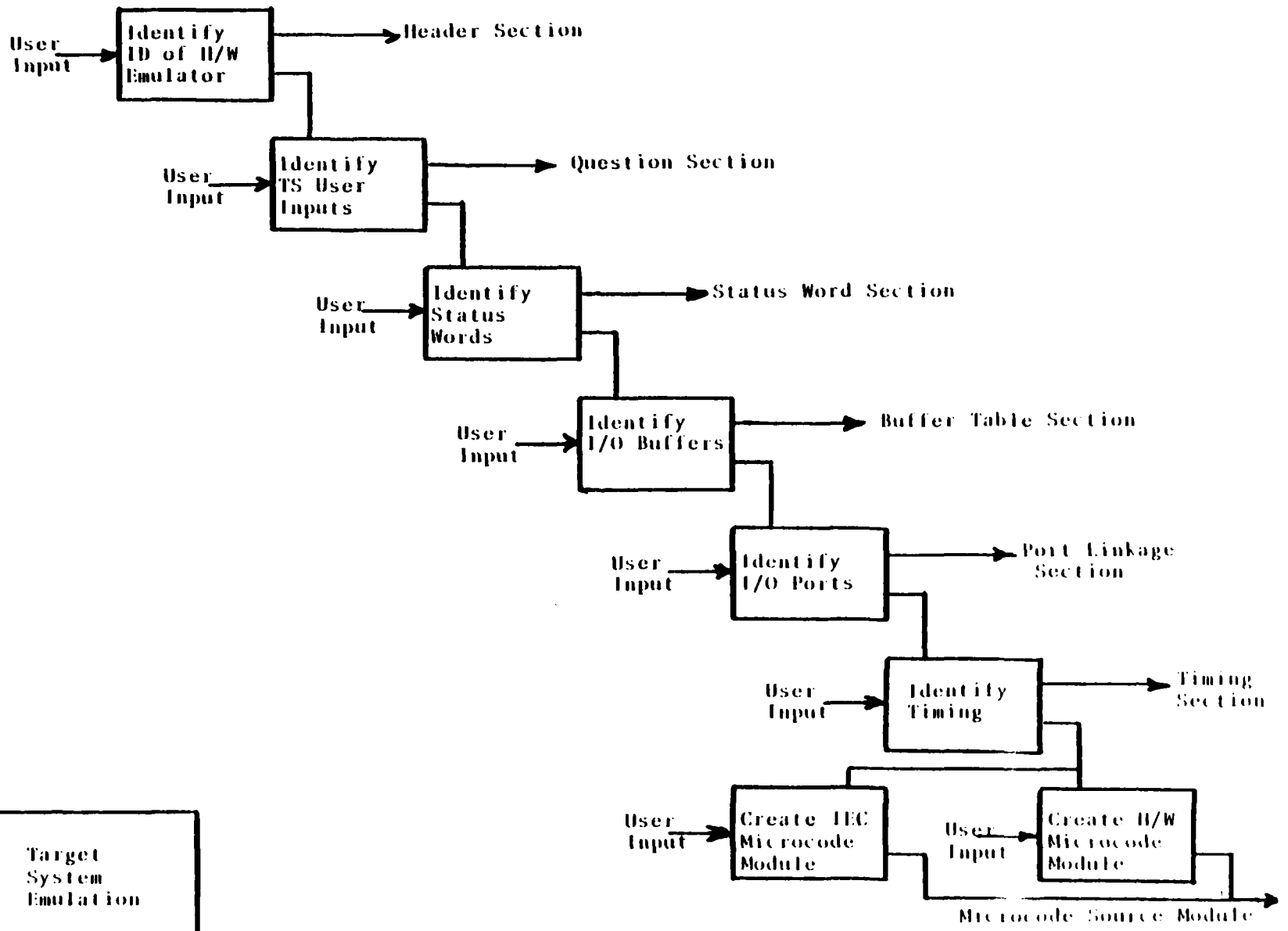
Figure 20.1 Top Down Design Chart

IØ - TARGET SYSTEM EMULATION

The Target System Emulation is a series of actions by the microcode programmer. These actions result in a Symbol Table and microcode object module existing in the host system, which can be linked through SGL with other modules to create a Target System.

This emulation involves eight major steps. These steps are: Identify ID of H/W Emulator, Identify TS User Inputs, Identify Status Words, Identify I/O Buffers, Identify Timing, Create IEC Microcode Module, or Create H/W Microcode Module. This creates the source of the emulation module. The majority of the work required to carry out these functions is done by the micro-programmer rather than by software. However, the Microcode Assembler does take care of assembly of the microcode source module to create the Symbol Table, and assembled microcode object module.

Note: For any of the following tables that are being built for input to the assembler, the microcode programmer can have the general user specify any variables at SGL time. (See the section on Identifying TS User Inputs).



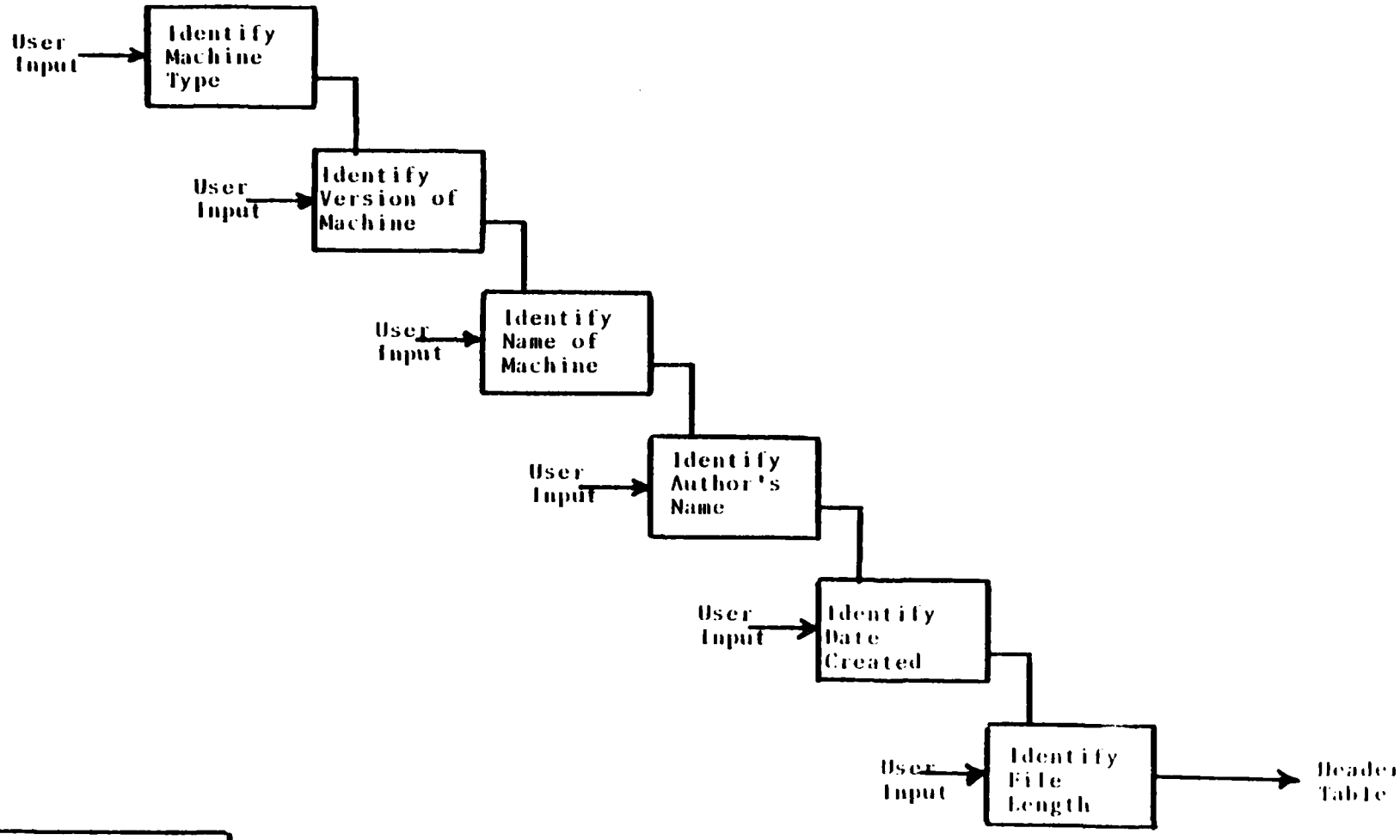
20-5

Iø Target System Emulation

11A - IDENTIFY ID OF H/W EMULATOR

The purpose is to identify, to the system and to the user, information concerning the module being created. This information might include the following descriptive data: Machine Type, Version of Emulator, Name of Machine, Author's Name, Date Created, and File Length. Once the format of this section is set, then all emulators will contain this information in the prescribed format.

All of this ID information is given by the programmer in the form of a Header Table. The programmer will place this information, in a standard format, at the beginning of the Target System Emulation Module. The Microassembler, when it assembles this Emulation Module, will also place this Header Table in the system directory which identifies all emulation modules. At this point, the assembler will add to the Header Table a reference to the storage of the emulation load module. This reference will probably be in the form of a disk file name.



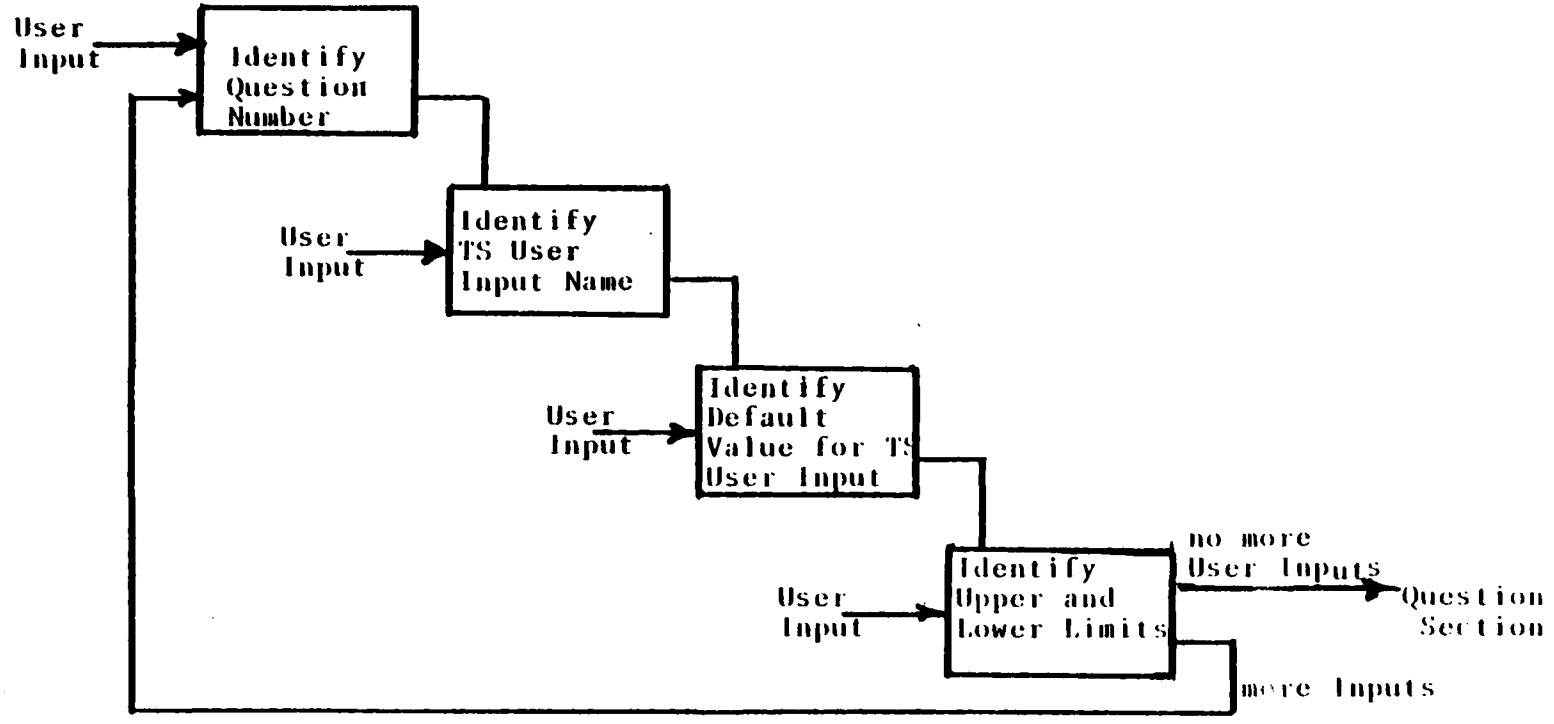
I1A Identify ID of H/W Emulator

11B - IDENTIFY TS USER INPUTS

Identifying User Inputs allows the user a certain amount of flexibility when a Target System is actually generated. Through SGL, the user will specify the system variables which are correct for his system. These system variables are identified to SGL by the creator of the microcode module. These variables make up the Question Section of the module.

The creator of the microcode module sets up the Question Section by identifying certain information for each system variable. This information is: the question number, the variable name, the default value, and the upper and lower limits. The question number might refer to a master table of questions maintained by SGL so that this section of the emulator will not have to be so large. The variable name corresponds to the name in the microcode module symbol table.

The Symbol Table, upon assembly of the microcode module, will contain references to these User Inputs . The variables will be flagged as User. Since the variables are only defined at SGL, an additional field in the Symbol Table refers to the address at which SGL should place the value of the symbol. The assembler handles any addressing indirectness which might occur.



6-03

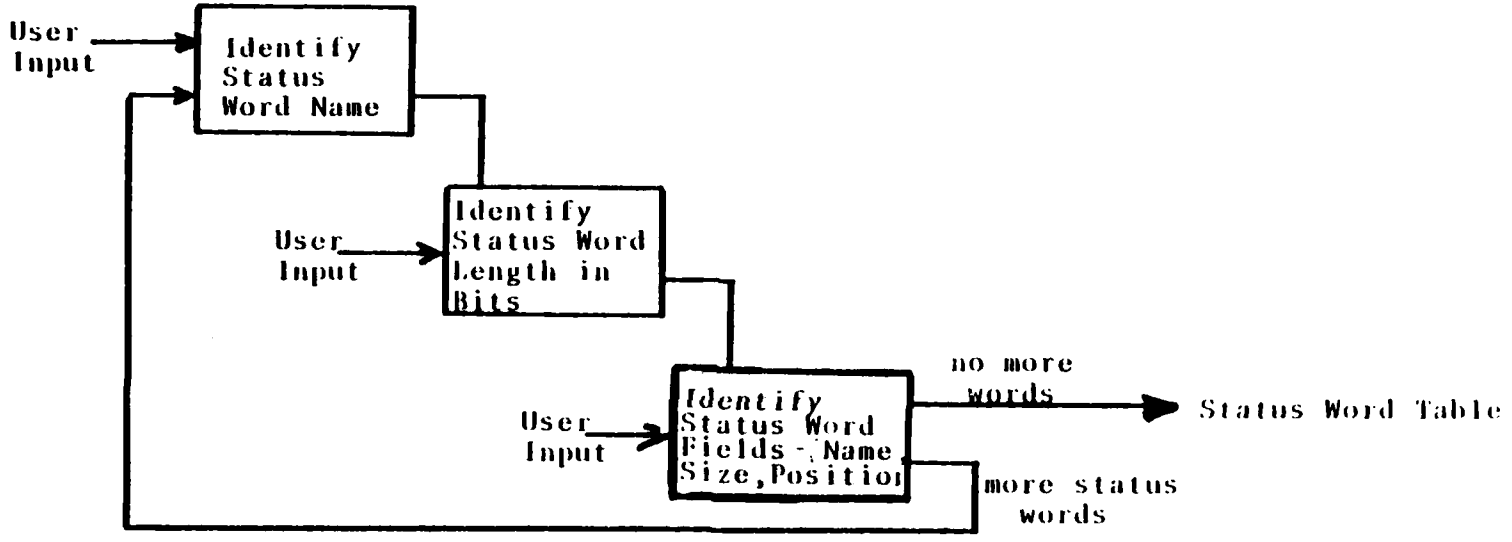
I1B Identify TS User Inputs

I1C - IDENTIFY STATUS WORDS

The purpose is to identify to the system all of the hardware status words which will exist as part of the emulation being created. These status words need to be made available to both the Target System and the microcode source module. This is accomplished by creating a Status Word Table which is placed in the microcode source module along with all other tables and microsubroutines.

The creator of the microcode module needs to specify certain information about each status word. This information includes: name of status word, size in bits, and number of fields in the status word. Then each field needed within the status word is identified by name, bit size, and its position in the status word.

As in the other tables, upon assembly by the Micro-assembler, all status words will be given a place in the Symbol Table which will include the name, address, and addressing information. The Status Word can also be a User-defined variable.



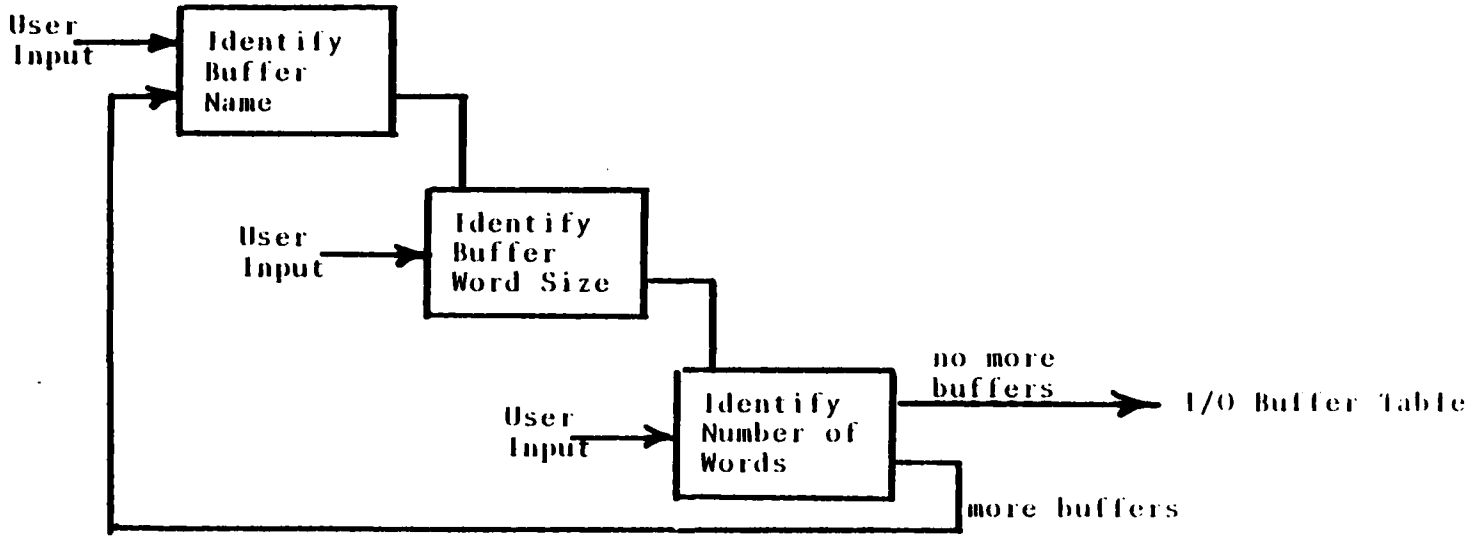
I1C Identify Status Words

11D - IDENTIFY I/O BUFFERS

I/O buffers need to be identified to the system to ensure that data that is input or output through a system device is stored in the proper place with the sufficient number of words.

The microcode module author needs to identify the name of the buffer, the word size of each word, and the number of words in the buffer. After he has identified all of the I/O buffers, he places the I/O Buffer Table into the microcode module.

During assembly, the I/O Buffer Information will be added to the Symbol Table. Then, an emulator's I/O buffers will exist in the PE's Internal Store. The address is stored in the Symbol Table along with each buffer entry.



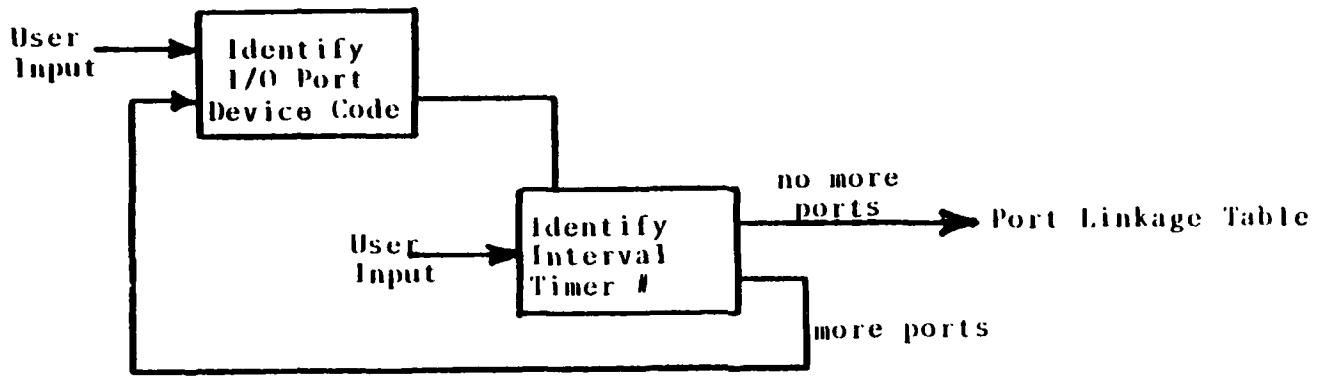
I 1D Identify I/O Buffers

I1E - IDENTIFY I/O PORTS

The I/O Ports need to be identified to the system so that other emulated hardware can communicate with this microcode module. The information supplied by the author is assembled in the form of a Port Linkage Table.

There are two sections to this table: I/O Port Device Code and the space for assigned Interval Timer Number associated with this port. The I/O Port Code associates the address in the Symbol Table at assembly time with the code of the I/O Port in the microcode source module. The I/O Port Device Code is supplied by the device manufacturer, and is inserted by the microcode programmer. The interval timer number is assigned by SGL.

The microassembler will build the Symbol Table and resolve the addressing in control store for the ports identified.



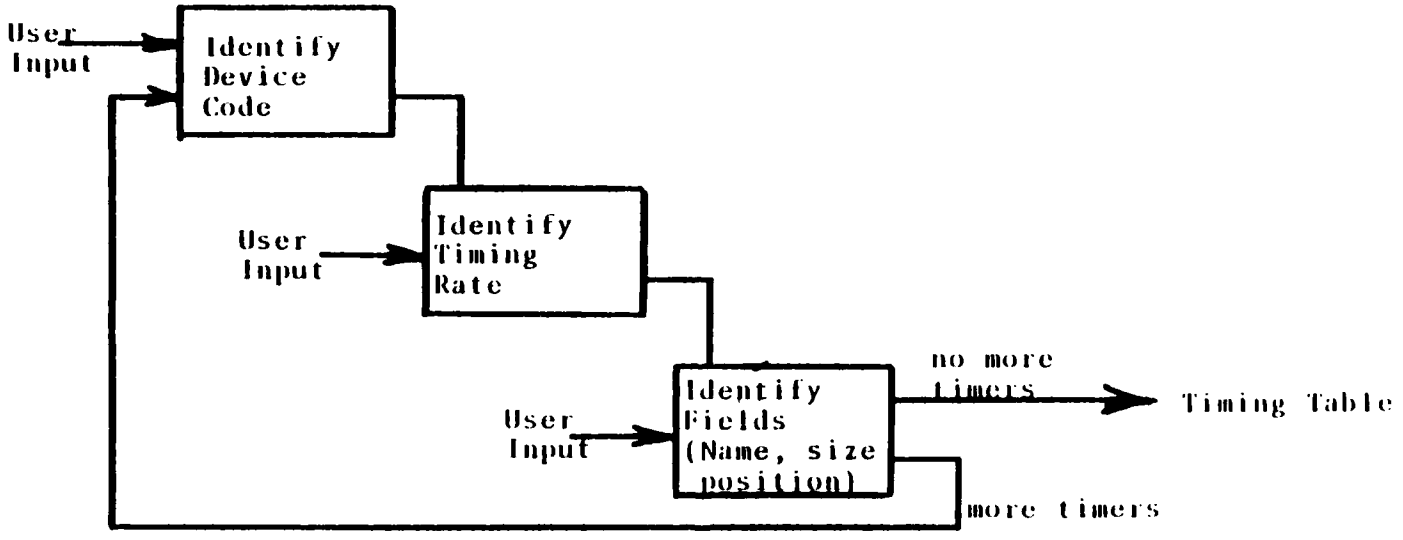
C-13

I1E Identify I/O Ports

I1F - IDENTIFY TIMING

The timing of the microcode module needs to be identified to the system in order to synchronize timing between H/W emulators. A timer is identified by Device Code, Rate, and associated fields of timer word. Each timer is also identified by a timer number.

The builder of the emulation module specifies in the Timing table the device code and the rate. The assembler resolves all references to the timing through building of the Symbol Table.

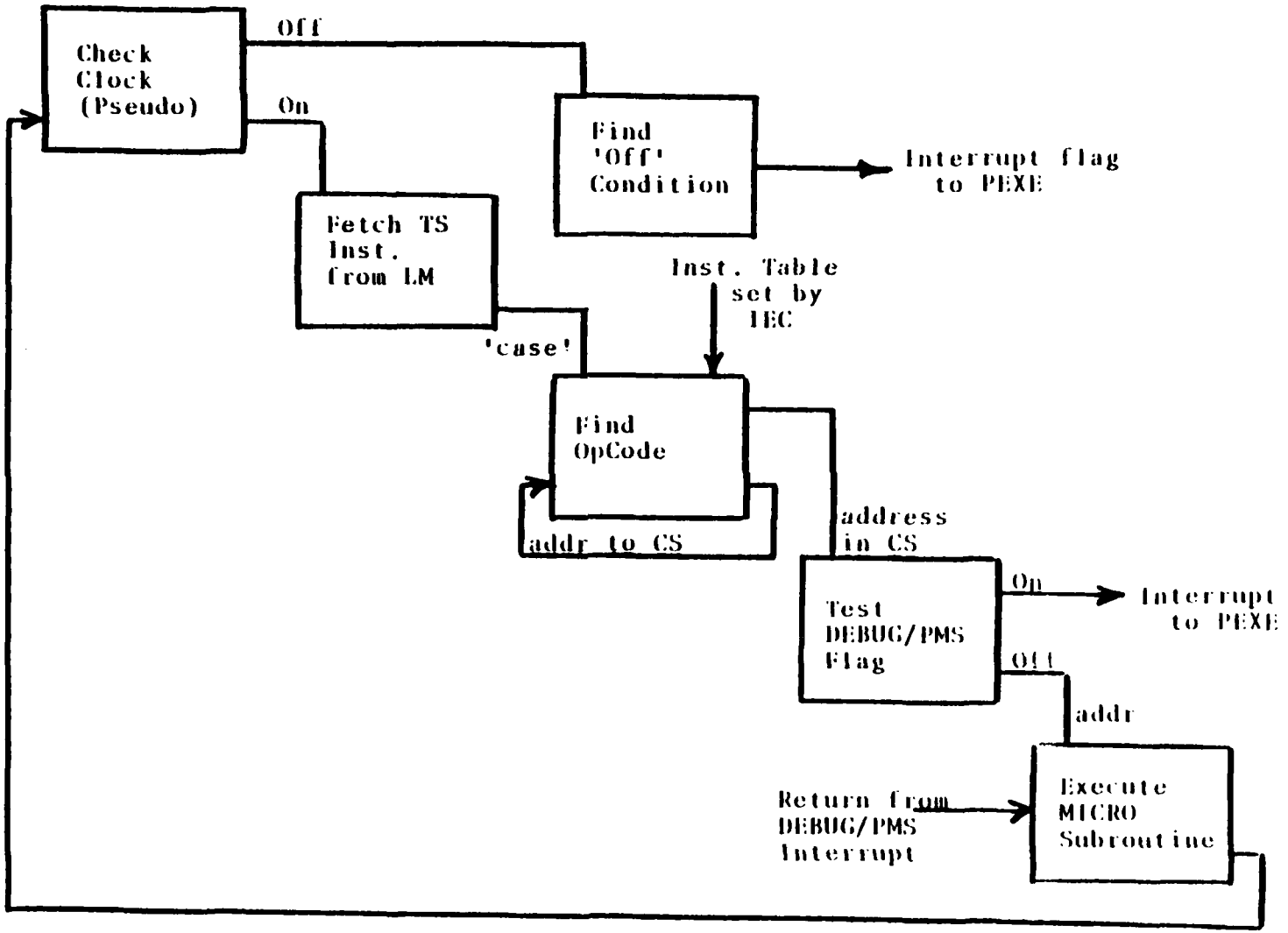


I1F Identify Timing

11G - CREATE IEC MICROCODE MODULE

The generation of an IEC microcode module is needed in order to emulate a particular Taget System instruction set. This IEC module needs to perform many functions. These functions are repeated in sequence: fetch TS instruction from the Load Module, decode instruction to obtain the Opcode, and execute the appropriate microsubroutines.

The creator of the microcode module needs to write microcode which will perform the above operation. This code, in a descriptive format, is appended to the tables that have previously been created. The Microassembler will assemble the entire microcode source module and produce a microcode object module and its Symbol Table. This output of the assembler is suitable for use by SGL.

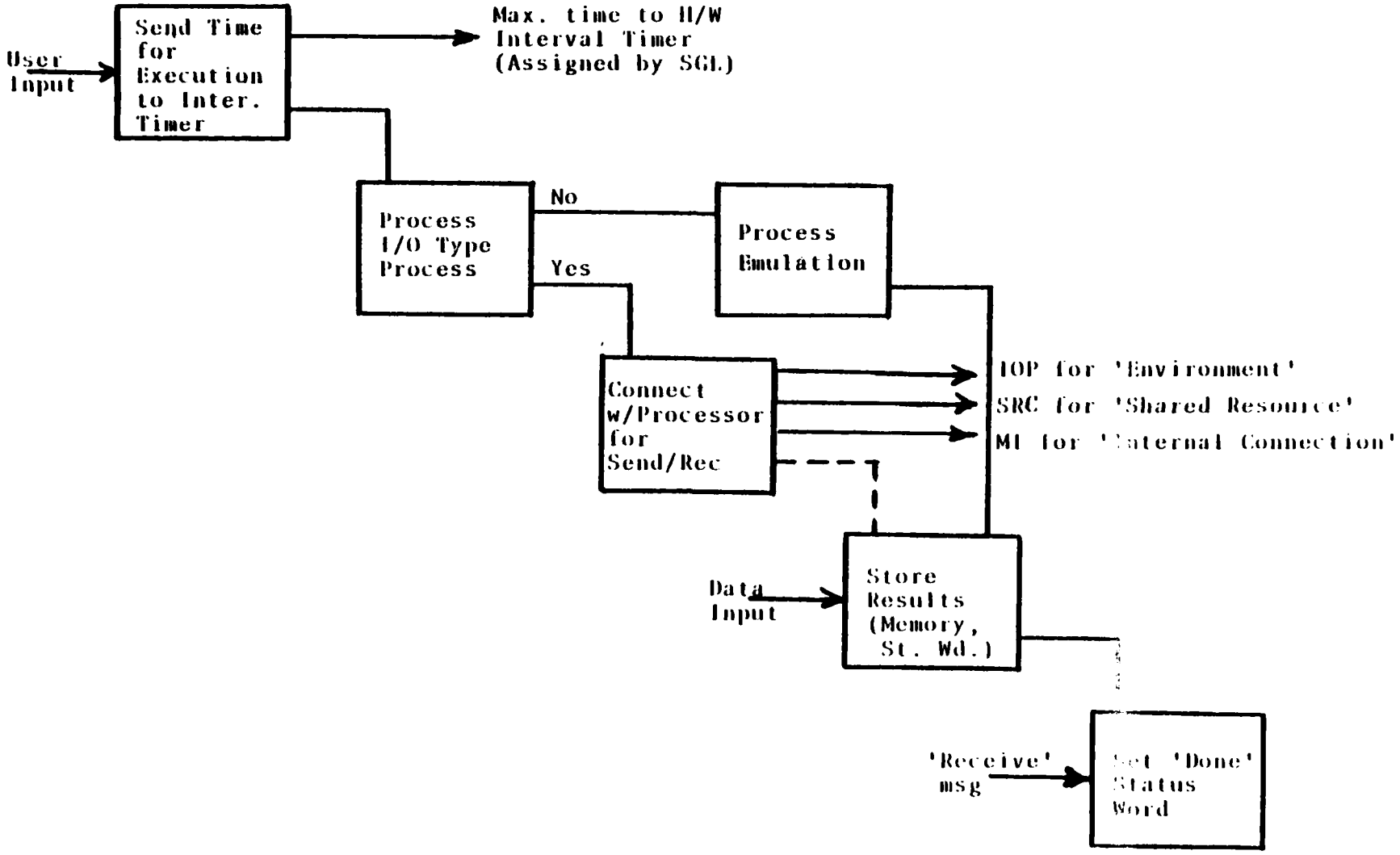


I1G Create IEC Microcode Module

I1H - CREATE H/W MICROCODE MODULE

Hardware microcode modules are used to emulate a particular hardware device. The emulation involves handling actual I/O processes as well as the emulation of I/O. The H/W microcode module must be able to distinguish between the two, to execute the correct microcode, and to store the results in the correct memory or status word address.

The microcode is written in the same microassembler code as the IEC module. This H/W module is then ready for assembly by the Microassembler.



I1H Create H/W Micro-code Module

20.2 PROPOSED TARGET SYSTEM

The previous section (20.1) described the needed information within every emulation module. The information has to follow some prescribed format in all cases, and one format was suggested.

A Target System (TS) consists of one or more Computer Systems (Cs). Each Cs will have one emulation module for the instruction set, which is called the IEC (for Instruction Emulation Code). The IEC will include the description of the word length, registers, memory size, etc. The needed I/O units are individually emulated, and the detail of these emulations depend upon the reason for the emulation. Thus, if the emulator of the disk is to study data transfer without regard to the placement of that data on the disk, then the emulation doesn't need the added detail on latency times, access times, etc.

The proposed Target System to be emulated involves two Intel 8080's and a PDP-11/45, which are able to communicate to each other over a bus. The two 8080's share memory. The PDP-11/45 has two perpheral devices: a magnetic tape drive and a disk drive. (See Figure 20.2).

There are two main groups to this emulation: 8080 modules and 11/45 modules. The 8080 modules include the Instruction Emulation Code, the Shared Memory emulation, and the emulation of the interprocessor communications. The 11/45 modules include the Instruction Emulation Code, the disk drive and controller, the magnetic tape drive and controller, and the emulation of interprocessor communications.

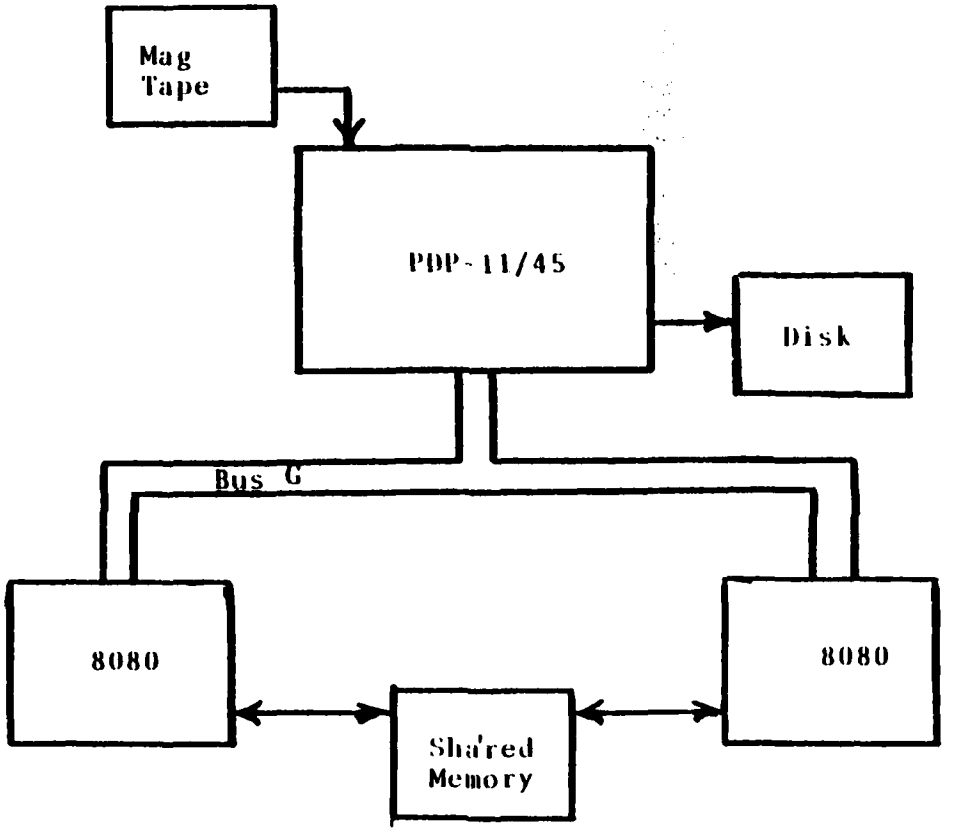


Figure 20.2 TS To Be Emulated

The disk and tape units on the 11/45 will interface to the FCP Computer through the IOP (I/O Processor). The shared memory between the two 8080's will interface through the SRC (Shared Resource Controller). The emulated bus between the PDP-11/45 and the two Intel 8080's is a simplified bus which illustrates one possible bus function; it is called "Bus G" to identify it as a non-standard device.

Each emulation module also requires a diagnostic package which the user utilizes to verify the correct utilization of that emulation module. "System Generation Loader" (Section 16) and "User Scenario" (Section 21) further discusses the purpose and approach to these test packages.

20.3 Emulation of PDP-11/45

The Target System involves one PDP-11/45 on a bus with two Intel 8080's. The 11/45 utilizes a magnetic tape as input and a disk device for output. These were selected to illustrate how to model these well-utilized peripherals, and also how to interface the TS with files of the FCP through the ICP. The bus between the 11/45 and the 8080's is designed so that the 11/45 is the Master and the 8080's are the slaves.

20.3.1 Emulation Modifications

Due to the way the PDP-11/45 handles peripheral devices, two routines need to be added to the Instruction Fetch cycle. These routines are the device interrupt handler and the device operation handler.

Each peripheral device has an interrupt vector address which can not be changed by software. This interrupt vector points to the location in memory which contains the interrupt service routine. Following the interrupt/vector/memory location is the device interrupt priority. Within the emulation, each device has an interrupt flag which is turned on when the device wishes to interrupt the processor. During the Fetch cycle when any flag is detected on, the priority of that interrupt is checked to see if it has a higher priority than the processor. If the priority is higher, control is passed to the appropriate interrupt service routine; otherwise, the interrupt is ignored.

The Interrupt Service Routine (ISR) is part of the PDP-11/45 applications software. Before control is given to the ISR, the Program Counter and Processor Status Word are saved on the processor stack.

The device operation handler is needed to start the peripheral device emulation. All status registers for the devices reside in the Local Memory of the PE which contains the PDP-11/45 IEC. The device handler checks the appropriate status bits for turning on the emulation. These bits are set by the applications software. Starting the emulation involves interrupting PEXE which, in turn, starts executing the device emulation (which may reside in a separate PE).

20.3.2 Instruction Emulation Code (IEC)

The IEC includes the ID Section, User Input Section, User Register Section, the "Fetch" Routine, Data Area, the symbolizing of all instructions, and the User-Defined Data Section. Figure 20.3.2-1 illustrates the details of three of these sections. Figure 20.3.2-2 illustrates the details of the emulation of some of the assembly commands of the 11/45.

ID Section

```
PDP 11/45
VØ1
Tom Jones
05/09/79
```

User Input Section

```
Q #, MSIZE, #, #
```

User Register Section

```
REGISTER GROUP

     REG, 16, 16

     SP, 16, 3

REGISTER,

     MAR(16),

     PSW(16),

PC = REG + 8
REGD = REG
REGS = REG
```

Figure 20.3.2-1 Beginning Details of the IEC for PDP-11/45

```
MOV:    TIMER← 300                  ;Move source to destination
        If(TEMPS.LT.Ø.OR.TEMPD.LT.Ø) GO TO MOVØ
        MAR←TEMPS                   ;get source address
        TEMP←M(MAR)                 ;get source
        MAR←TEMPD                   ;get dest. address
        N←SIGNI6                    ;put source in dest.
        Z←ZERO                      ;       "
        V←Ø                         ;       "

        GO TO FETCH

MOVØ:   If(TEMPD.LT.Ø.AND.TEMPS.GE.Ø) GO TO MOVD
        If(TEMPD.GE.Ø.AND.TEMPS.LT.Ø) GO TO MOVS
        R(REGD)←R(REGS)             ;put source reg. in dest. reg.
        N←SIGN16                    ;set condition flag
        Z←ZERO                      ;       "
        V←Ø                         ;       "

        GO TO FETCH

MOVD:   MAR←TEMPS                   :Get source address
        R(REGD)←M(MAR)              ;put source in dest. reg.
        N←SIGN16                    ;set condition flag
        Z←ZERO                      ;       "
        V←Ø                         ;       "

        GO TO FETCH

MOVS:   MAR←TEMPD                   ;Get dest. address
        M(MAR)←R(REGS)              ;put source reg. in dest.
        N←SIGN16                    ;set condition flag
        Z←ZERO                      ;       "
        V←Ø                         ;       "

        GO TO FETCH

CMP:    TIMER←300                   ;Compare source and destination
        If(TEMPS.LT.Ø.OR.TEMPD.LT.Ø) GO TO CMPØ
        MAR←TEMPS                   ;get source address
        TEMP←M(MAR)                 ;get source
        MAR←TEMPD                   ;get dest. address
        TEMP2←M(MAR)                ;get destination
        TEMP←TEMP-TEMP2             ;compare source and dest.
        N←SIGN16                    ;set condition flag
        Z←ZERO                      ;       "
        V←INTOVFL                   ;       "
        C←CARRY16                   ;       "

        GO TO FETCH
```

Figure 20.3.2-2 Emulation of Some PDP-11/45 Commands

20.3.3 Disk Configuration

The configuration of the RSO4 is one which requires no variables for the emulation. There is one disk drive. On this disk are 32 tracks, 64 sectors per track, and 32 16-bit words per sector. This amounts to 64K words of storage.

There are eight device registers which are used to handle all commands and status for disk I/O. A description of all of the registers and their locations in memory follow. The bits of the registers which are not used in the emulation are kept at "0".

Address	
777440	Look Ahead Register (RCLA)
777442	Disk Address Register (RCDA)
777444	Disk Error Status Register (RCER)
777446	Disk Control and Status Register (RCCS)
777450	Word Count Register (RCWC)
777452	Current Address Register (RCCA)
777454	Maintenance Register
777456	Data Buffer Register (RCDB)

Disk Emulation Routine

The emulation process for disk is simple. At Initialization of MMS during RTE, the appropriate status bits are set. During execution of the TS, the emulation routine for disk looks at the functions bits to determine what operation is to take place, Read or Write. Another two functions exist, but are meaningless without the actual hardware; these are, therefore, ignored. PEXE is then sent an interrupt to indicate whether a Read or Write is to take place, the track and sector on the disk and the memory location in the IEC's Local Memory of where to retrieve or store the data. An I/O Handler within PEXE handles the actual operation and then returns control to the emulation code. Status and error bits are set and all data is operated on until the Word Count is zero.

If the Interrupt Enable bit is set, the DIFLAG is turned on in the IEC. Execution of the emulation code stops. The IEC then acts according to which status bits are set.

20.3.4 Magnetic Tape Configuration

The configuration of the TU10 requires no variables for the emulation. There is one mag tape unit. In this emulation, there is no difference between 7-track or 9-track tapes. Information is stored in 8-bit bytes, and there is a capability for storage of 20 million bytes.

There are six device registers which are used to handle commands and status for mag tape I/O. A description of all of the registers and their locations in memory follow. The bits of the registers which are not used in the emulation are kept at "0".

Address	
772520	Status Register (MTS)
772522	Command Register (MTC)
772524	Byte Record Counter (MTBRC)
772526	Current Memory Address Register (MTCMA)
772530	Data Buffer (MTD)
772532	Two Read Lines (MTRD)

Mag Tape Emulation Routine

When the magentic tape emulation is started, appropriate status bits are set to indicate that the tape is in operation. The functions bits of Magnetic Tape Controller (MTC) are checked to determine which of eight operations is to be performed. During an operation, PEXE may be interrupted to perform the actual I/O and give the data back using a Direct Memory Access transfer. Status and any error bits are set when PEXE gives control back to the emulation.

If the Interrupt Enable bit is set, the MTIFLAG is turned on in the IEC. Execution of the emulation code then stops, awaiting startup by PEXE.

The application program then acts accordingly, depending on the status of the device registers.

20.3.3 PDP to Intel Communication Configuration - "Bus G"

There are two facets to the PDP's communication with the 8080: PDP initiating a function and the 8080 initiating a function. In both cases, I/O transfers are one 8-bit byte at a time.

The PDP initiates a function by placing the appropriate data in the "Address" and "Command and Status" Registers. Also, the GO bit is set. During the PDP IEC's instruction fetch cycle, the Ready and GO bits are checked. If they are on, an I/O function is initiated which operates between the PDP's memory and its I/O port. Only when the 8080 lets the PDP know that it wants to act on the data is data actually moved between PE's.

The 8080 initiates an I/O operation by setting a flag. This flag is checked by the PDP during its fetch cycle. If the flag has been turned on, the 8080's Operation ID is checked and servicing is initiated.

The two registers utilized, then, within the communication configuration are:

Address	
777170	Command and Status Register (INST)
777172	Address Register (INAD)

Thus, "Bus G" is not designed after any standard bus, but is an emulation designed to accomplish communication between the PDP-11/45 and the two Intel 8080's.

20.4 Emulation of Intel 8080

The Target System involves two 8080's. The purpose is to show how one set of emulation code can be inserted twice in order to simulate the two computers. Both are identical: IEC and a memory unit shared by the two 8080's. Both computers are also tied to a PDP-11/45 as "slaves" to that "master".

20.4.1 Instruction Emulation Code (IEC)

The IEC involves the ID Section, User Input Section, User Register Section, the "Fetch" Routine, Data Area, the symbolizing of all instructions, and the User-Defined Data Section.

Figure 20.4.1-1 illustrates the details of the first sections of the IEC. Figure 20.4.1-2 shows the details of the FETCH Routine. Figure 20.4.1-3 illustrates the details of some of the 8080 commands.

```
ID Section
Intel 8080
VØ1
TOM JONES
05/09/79

User Input Section
QN, MSIZE, 64, 8, 64

User Register Section
REGISTER GROUP,
     SR1, 8, 8;
REGISTER,
     SP(16),
          SPH = SP(15-8),
          SPL = SP(7-Ø);
     PC(16),
          PCH = PC(15-8),
          PCL = PC(7-Ø);
     MAR(16),
          MARH = MAR(15-8),
          MARL = MAR(7-Ø);
SR2 = SR1
SR3 = SR1
```

Figure 20.4.1-1 Beginning Details of the IEC for INTEL 8080

FETCH ROUTINE

```
FETCH:          IF(PCLOCK.EQ.1) Interrupt PEXE for COFF
          <------all I/O Handler code ------>
                IR←M(PC)
                PC←PC+1
                DBGPMS←D31D30D29
                IF(DBGPMS.EQ."COND") Interrupt PEXE
                CASEIN←IR
CASE Ø:         CASE<7:6>  6,0
                GO TO @(CASER)
CASE 1:         CASE<2:0>  0,4
                GO TO @(CASER)
CASE 2:         CASE<5:3>  3,12
                GO TO @(CASER)
CASE 3:         CASE<5:3>  3,20
                GO TO @(CASER)
CASE 4:         CASE<5:3>  3,28
                GO TO @(CASER)
CASE 5:         SR1←D5D4D3
                SR2←D2D1D0
                IF(SR1.EQ.'110'.AND.SR2.EQ.'110) GO TO HALT
                GO TO MOV
CASE 6:         SR1←D2D1D0
                CASE<5:3>  3,36
                GO TO @(CASER)
CASE 7:         CASE<2:0>  0,44
                GO TO @(CASER)
CASE 8:         CASE<5:3>  3,52
                GO TO @(CASER)
CASE 9:         CASE<5:3>  3,60
                GO TO @(CASER)
CASE 10:        CASE<5:3>  3,68
                GO TO @(CASER)
CASE 11:        CASE<5:3>  3,76
                GO TO @(CASER)
CASE 12:        CASE<5:3>  3,84
                GO TO @(CASER)
CASE 13:        CASE<5:3>  3,92
                GO TO @(CASER)
CASE 14:        CASE<5:3>  3,100
                GO TO @(CASER)
```

Figure 20.4.1-2 INTEL 8080 FETCH Routine

```
DADLXI:   TIMER←1500                    ;Set up for DAD or LXI
          SR2←IR(5-3)                   ;Get code for either inst.
          If(SR2 is ODD) GO TO DAD      ;Check for DAD

LXI:                                    ;Load register pair immediately
          SR2←SR2+1                     ;set for low-order of reg. pair
          R(SR2)←M(PC)                  ;store low-order data in reg.
          SR2←SR2-1                     ;set for high-order of reg. pair
          PC←PC+1                       ;Up PC for high-order data
          R(SR2)←M(PC)                  ;store high-order data in reg.
          GO TO FETCH

DAD:                                    ;Add register pair to H and L
          SR1←101                       ;set for register L of reg. pair
          R(SR1)←R(SR1)+R(SR2)          ;add low-order reg. to reg. L
          SR1←110                       ;set for register H of reg. pair
          SR2←SR2-1                     ;set for high-order reg. of pair
          R9(SR1)←R(SR1)+R(SR2)+CARRY8  ;add high-order reg to H + carry
                                           ;from previous low-order add
          CY←CARRY8                     ;set carry for 8080 code
          GO TO FETCH

DCXINX:   TIMER←500                     ;Set up for DCX or INX
          SR2←IR(5-3)                   ;get reg. pair in bits 5-4
          If(SR2 is odd) GO TO DCX      ;bit 3 on means DCX

INX:                                    ;Increment register pair
          SR2←SR2+1                     ;get low-order of register pair
          R(SR2)←R(SR2)+1               ;increment low-order of reg. pair
          SR2←SR2-1                     ;set for high-order of reg. pair
          R(SR2)←R(SR2)+CARRY8          ;increment high-order if carry
                                        ;is set
          GO TO FETCH

DCX:                                    ;Decrement register pair
          R(SR2)←R(SR2)-1               ;decrement low order of reg. pair
          SR2←SR2-1                     ;set for high-order of reg. pair
          R(SR2)←R(SR2)-CARRY8          ;decrement high-order if carry
                                        ;is set
          GO TO FETCH

INR:      TIMER←500                     ;Increment register
          SR1←IR(5-3)                   ;set for register
          IF(SR1.EQ.110)GO TO INRM      ;check for memory ref.
          R(SR1)←R(SR1) + 1             ;increment register
          Z←ZERO                        ;set condition flag
          S←SIGN8                       ;    "      "
          P←PARITY                      ;    "      "
          AC←CARRY4                     ;    "      "
          GO TO FETCH
```

Figure 20.4.1-3 Details of Some INTEL Commands

20.4.2 Shared Memory

The emulation of shared memory allows the Intel 8080 to share memory with one or more processors. The emulation requires one module and the use of the Shared Resource Controller (SRC). This module is the Shared Memory Interface Module (SMIM) which resides in the PE. The SRC allocates the use of the shared memory to those PE's which attempt to use it.

The SMIM performs the actual I/O operation between the PE referencing the shared memory and the shared memory itself. During SGL, the appropriate control blocks are built for storage in the SRC. These control blocks contain information about the scheduling of the shared resource to the PE's. In addition, SGL supplies to the module information concerning the location of the shared resource.

Upon execution of a memory reference instruction, the Memory Map Table determines if the memory is local or shared. If it is shared, the hardware sets a flag indicating the need for a shared memory reference. The microcode associated with the memory operation checks this flag. If the flag has been set, a call is made to SMIM.

This module, through bus communications, sends a message to the SRC asking for use of the shared memory. When the SRC determines that the shared memory is available for use, it sends a "request granted" message to SMIM. The SMIM then performs the appropriate function, read or write, using the shared memory.

When the operation has been completed, SMIM sends a release message to the SRC and then returns control to the IEC.

20.4.3 Intel 8080 to PDP-11/45 Communication - "Bus G"

The only device with which the 8080 processor communicates is the PDP-11/45. Therefore, the IN and OUT instructions can be dedicated to serving one device. When the 8080 wishes to talk with the 11/45, it sets the flag WANT. The 11/45 is continuously (every Fetch cycle in IEC) polling this flag and when it sees that the flag is on, it services the "interrupt". When the operation requested by the 8080 has been completed, WANT is cleared. The 8080 recognizes this change and then continues processing.

Thus, "Bus G" is not designed after any standard bus, but is an emulation designed to accomplish communication between an Intel 8080 and the PDP-11/45.

20.5 Register Emulation

The registers of a processor are emulated so that they can be manipulated by the processors as if they were in their own machine. The emulation of the register may address them either as part of the PE's software registers, or as data words in the PE's Internal Memory. The emulation of both cases is similar.

Each register has a unique numeric value associated with it. This value is used by the register emulation to resolve addressing within a PE.

A Register Group is needed for each group of registers being emulated. This register group is identified by a variable in the User Register Section of an IEC module. When an emulated register needs to be referenced, its unique numeric value is placed in the register group variable. The register is then referenced with the R(<register group>) convention. The macrocode then resolves PE addressing by taking the address of the register group variable + 1, and adding it to the value in the variable itself. This yields either a unique internal memory address or a software register.

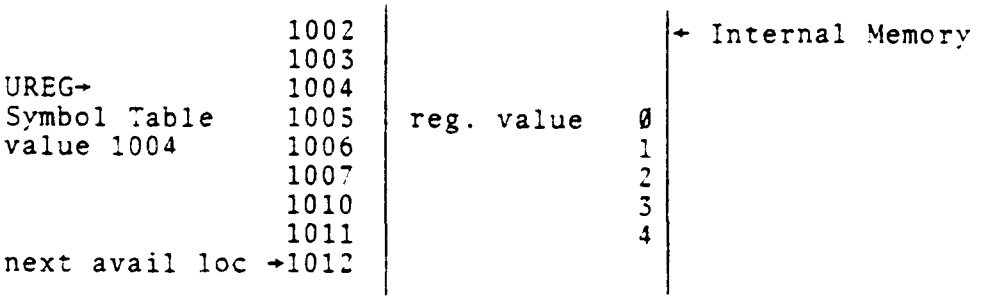
To illustrate, if the description of register UREG is part of the header "REGISTER GROUP", within the emulation:

```
REGISTER GROUP,          ;In User Register Section

   UREG,8,5              ;Register Group Name - UREG
                         ;referenced by UREG←I.D. Value
                         ;        R(UREG)←
                         ;size of register - 8 bits
                         ;number of register - 5 (Ø-4)
```

Then the emulation through Internal Memory would be:

Internal Memory Emulation



And, the emulation through Software Registers would be:

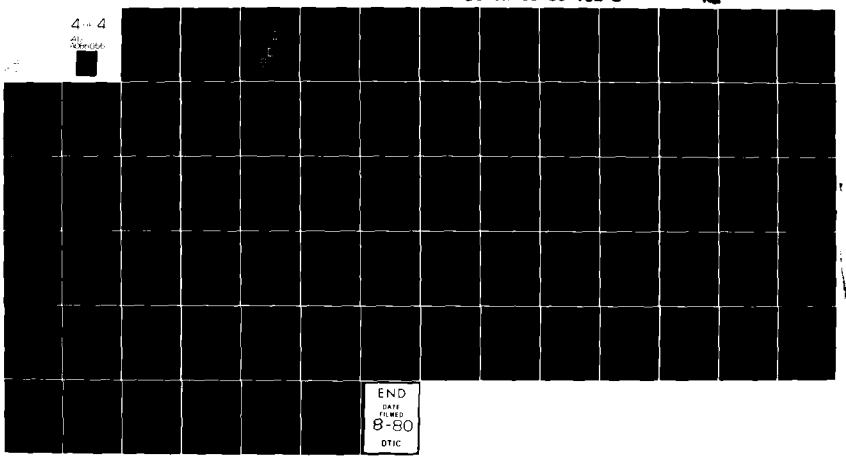
Software Register Emulation

UREG assigned	Software Register	4
register value	0	"	5
"	1	"	6
"	2	"	7
"	3	"	8
"	4	"	9

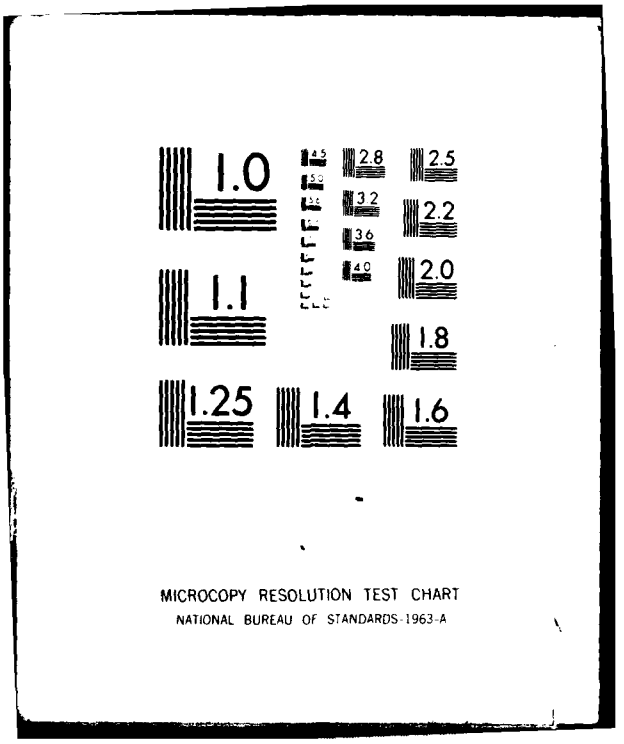
- Software Register 10 is next register available.

AD-A086 066 HARRIS CORP MELBOURNE FL GOVERNMENT COMMUNICATION SY--ETC F/G 9/2
MULTIPLE MICROPROCESSOR SYSTEM (MMS) DESIGN STUDY. VOLUME II.(U)
MAR 80 F30602-78-C-0114

UNCLASSIFIED RADC-TR-80-33-VOL-2 NL



END
DATE FILMED
8-80
DTIC


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

20.6 Abbreviations Used in Section 20

Cs	Computer system
CS	Control Store
H/W	Hardware
ID	Identification
IEC	Instruction Emulation Code
Inst.	Instruction
Inter.	Interval
IC	Input/Output
I/O	Input/Output
IOP	Input/Output Processor
ISR	Interrupt Service Routine
LM	Local Memory
Max.	Maximum
MI	Memory Interface Unit
MMS	Multiple Microprocessor System
Msg.	Message
MTC	Magnetic Tape Controller
MUX	Multiplexer
OpCode	Operation Code
PE	Processing Element
PEXE	PE Executive
PMS	Performance Monitoring System
Rec.	Receive
RTE	Run-Time Executive
SGL	System Generation Loader
SMIM	Shared Memory Interface Module
SRC	Shared Resource Controller
St. Wd.	Status Word Address
TS	Target System
w	with

21.0 USER SCENARIO

Sections 12 to 20 have presented the software necessary for MMS. The general user will directly utilize only a few of these packages:

SGL (System Generation Loader)

RTE (Run-Time Executive)

PMD (Post Mortem Dump)

PMS (PMS Off-Line Analysis)

The purpose of these packages have been explained several times (see Sections 11, 16, 17, 18, 19, and 20). In fact, Section 20 also discusses the structure of a proposed Target System to be utilized on MMS.

Since the above-named packages will be utilized by users of all level of experience, these packages have to be able to respond to these different levels. "Help" files have to be available. These programs have to be designed so that users can easily utilize them without having to learn detailed languages. The programs also have to be designed so that experienced users do not have to wait for CRT displays or for the FCP computer to accomplish the specified tasks. In the discussions and examples that follow, these users' needs are incorporated into the designs.

Figure 21.0-1, which is the same as Figure 11.0-3, shows the detailed modular design of MMS. The four programs, previously mentioned as being user oriented, are depicted with heavy boxes.

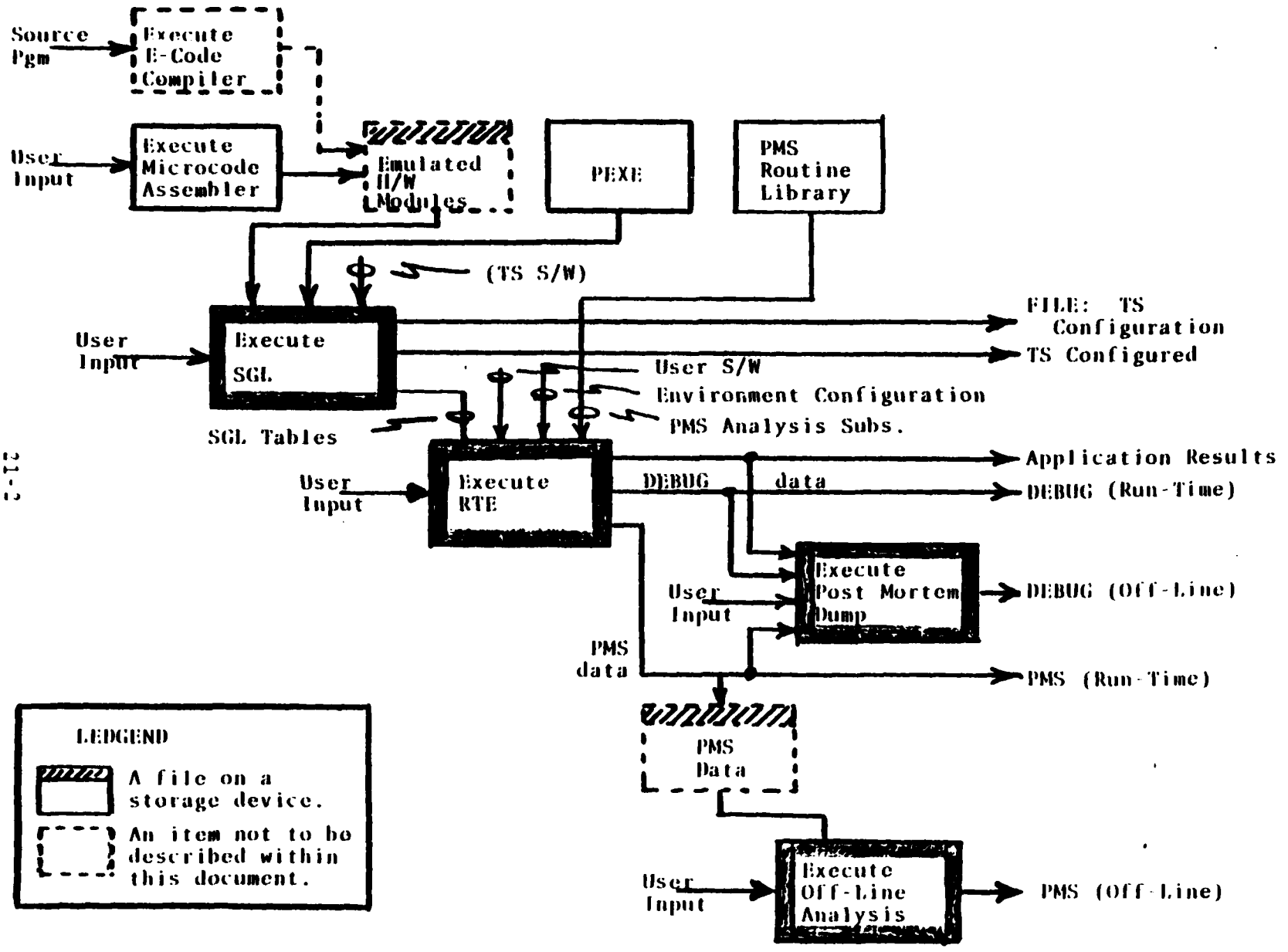


Figure 21.0-1 Detailed Modular Design of MMS

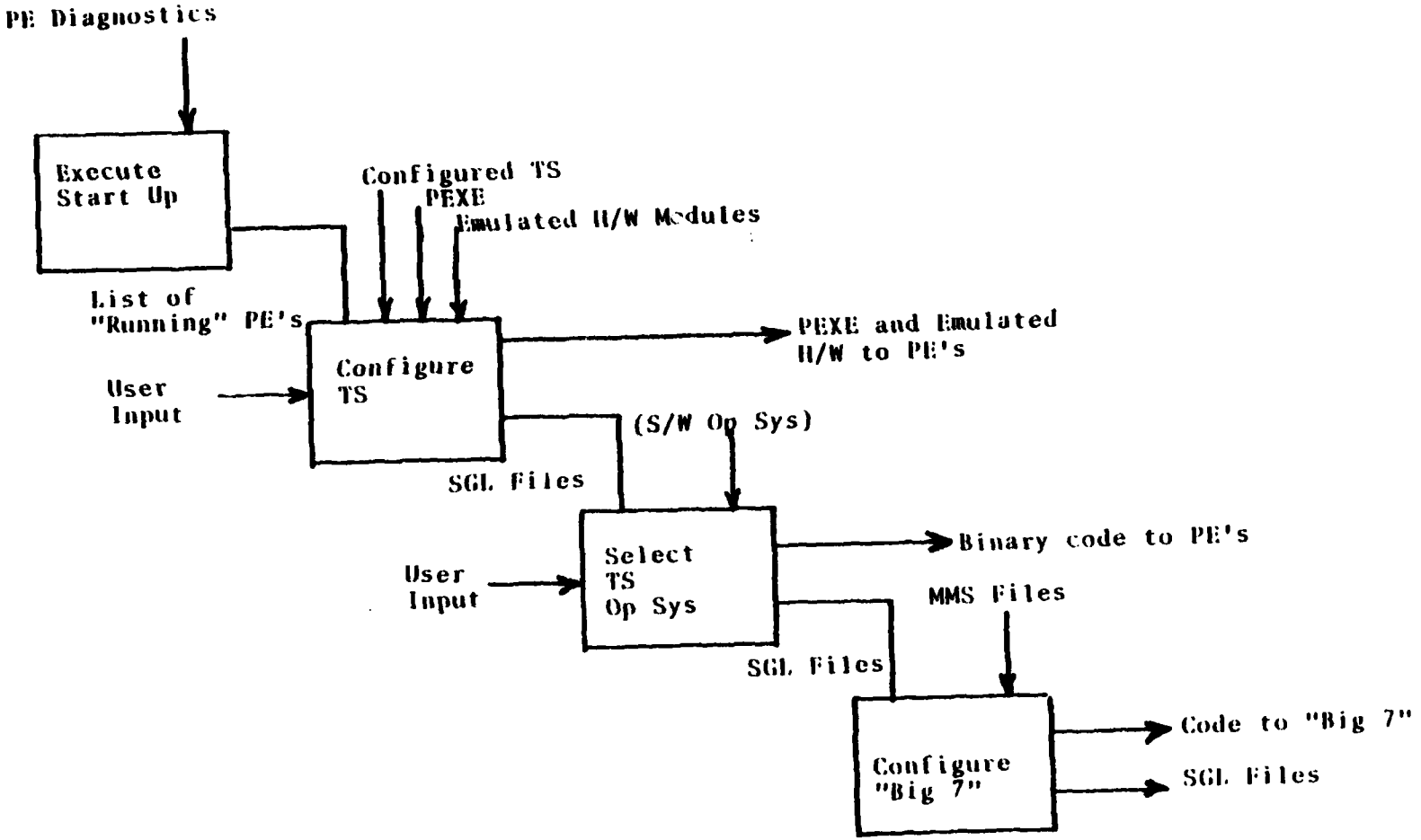
With reference to the program SGL (System Generation Loader), it can be located within Figure 21.0-1 as "Execute SGL". Figure 21.0-2. "EØ - MMS SGL", then shows more detail of SGL. This package, from the user viewpoint, will be discussed in Section 21.2.

Relative to RTE (Run-Time Executive), this is at "Execute RTE" in Figure 21.0-1 or, in more detail, as Figure 21.0-3, "FØ-Run-Time Executive (RTE)". The user's viewpoint will be discussed in Section 21.3.

The program "Post Mortem Dump" is found in Figure 21.0-1 as "Execute Post Mortem Dump", or as Figure 21.0-4 as "GØ - Post Mortem Dump". This program is discussed in Section 21.4.

The last user-oriented package, PMS Off-Line Analysis, is found in Figure 21.0-1 as "Execute Off-Line Analysis", or in Figure 21.0-5, which is "HØ - PMS Off-Line Analysis". This forms the discussion of Section 21.5.

Throughout all four packages, the command input for the interactive user is by selection to a printed menu. All possible responses are indicated. A response other than those indicated will produce a NOP (no operation). Whenever the user doesn't understand an option, he can get an explanation by asking for the display of the "Help" file. This "Help" file, common to the four user packages, is presented in Section 21.1.



EØ

MMS SGL

FIGURE 21.0-2 Top-level Design of SGL

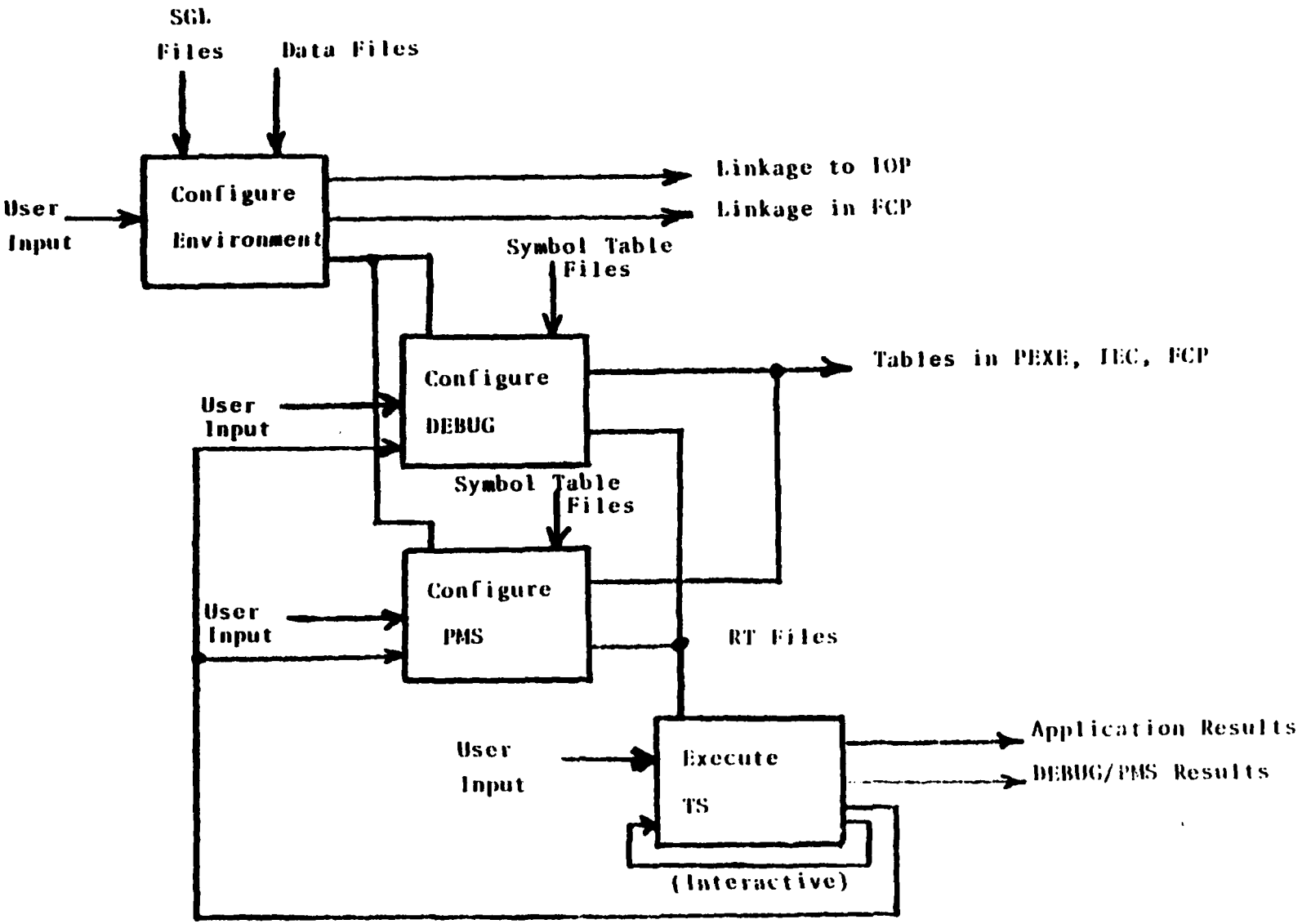
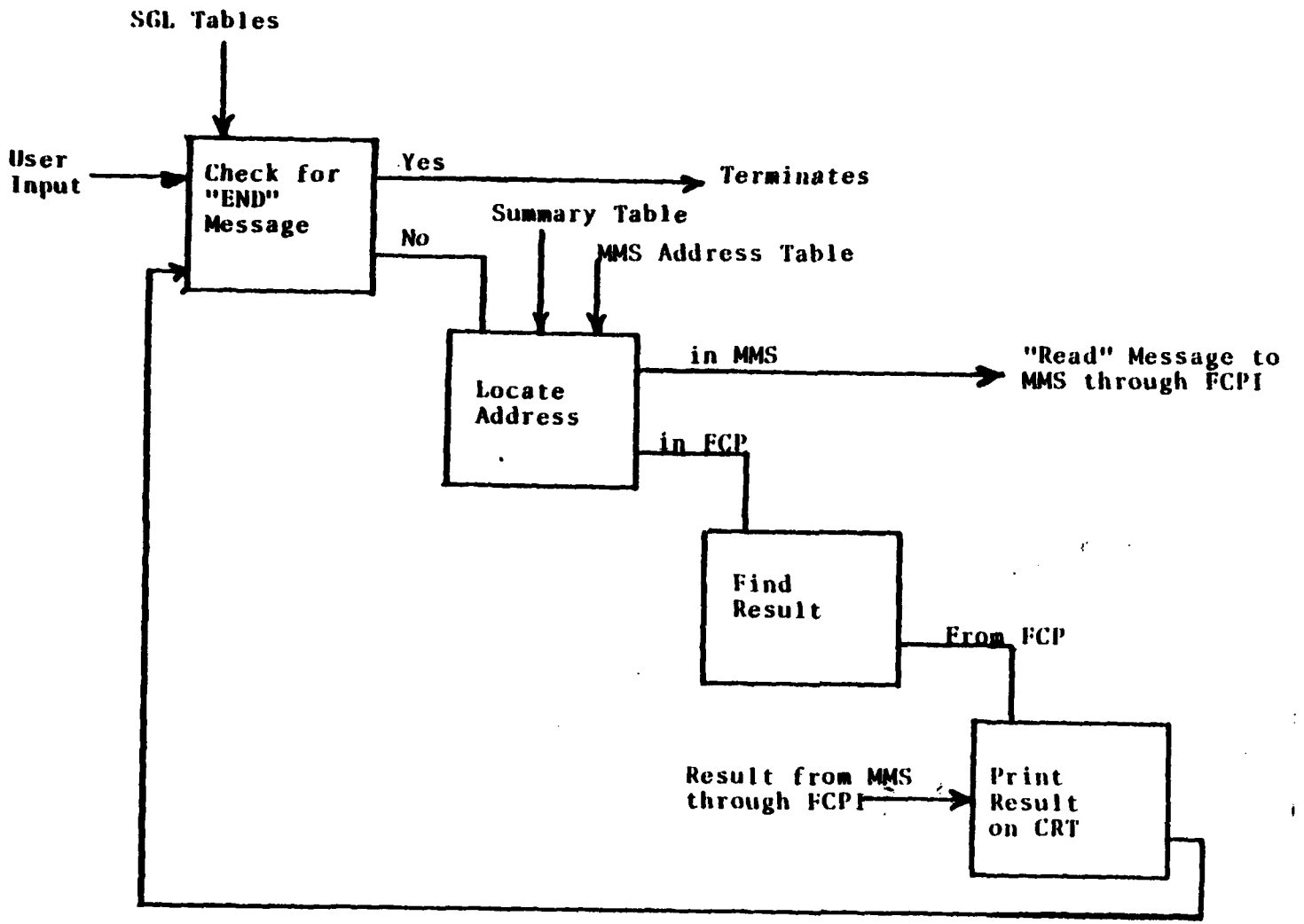
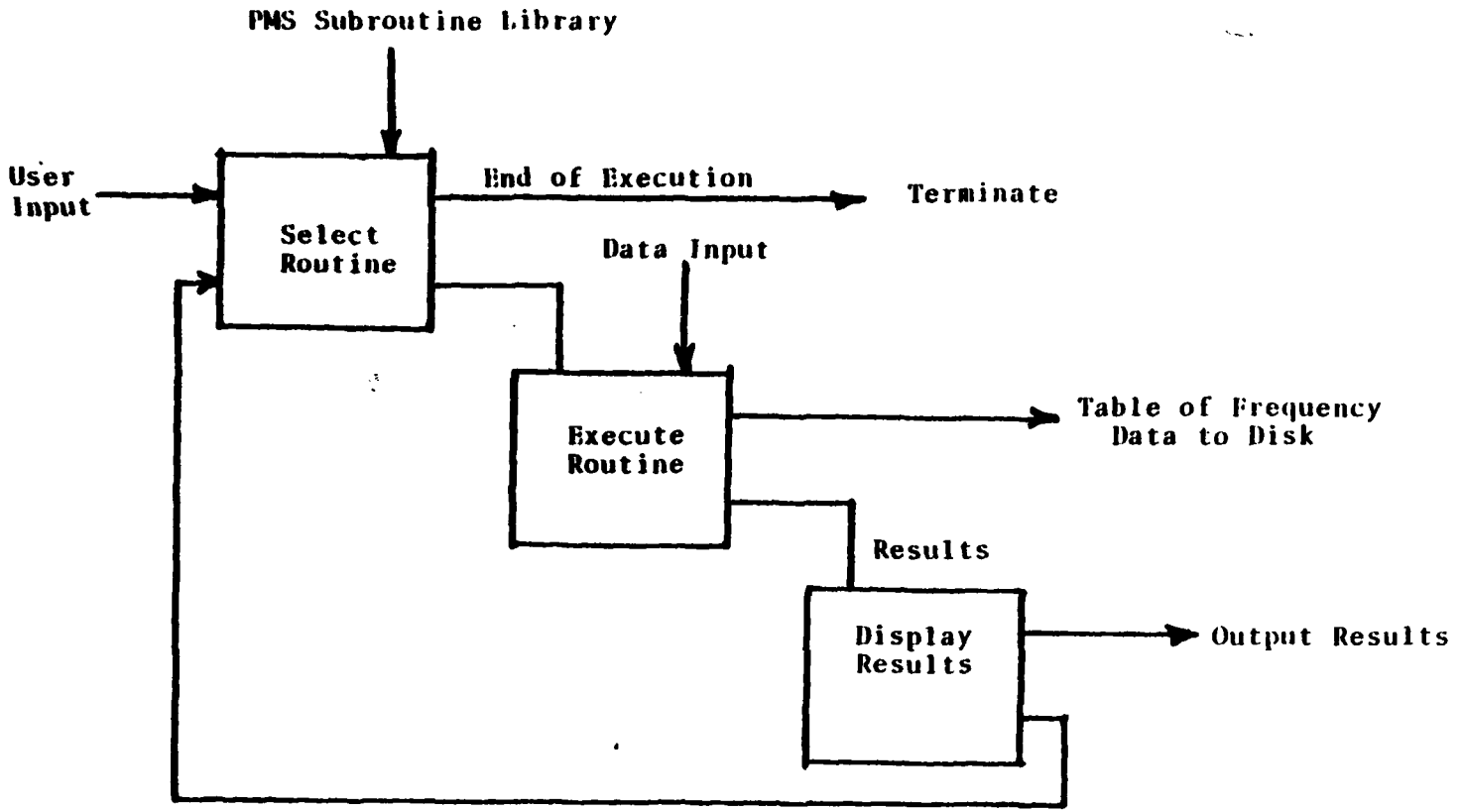


Figure 21.0-3 Top-level Design of RTE



GØ Post Mortem Dump

Figure 21.0-4 Top-level Design of PMD



H0 PMS Off-line Analysis

Figure 21.0-5 Top-level Design of PMS

21.1 "Help" File

The "Help" file can be displayed at any time by the interactive user, who is a user involved with executing a program from the CRT. The selection of options can be by the full word or by the first initial only, because all options available to the user have separate initial letters.

Figure 21.1. lists the options of the "help" file. A further explanation of each option is part of this section. Alphabetically, the list is:

C or CONTINUE - Whenever the user chooses to stop one of these software packages, he can input P or PAUSE. Then, if the user opts to continue the normal usage of the software package, he inputs this input of C or CONTINUE. The P | PAUSE did not distroy the original display; the C | CONTINUE also will not destroy the original display.

D or DONE - When the user is told that this option is selectable, he can choose to terminate the execution of the software package. If applicable, the tables and generated emulated systems are saved.

H or HELP - When the user decides that he doesn't remember the meaning of any of the options, a "Help" listing as depicted by Figure 21.1 is displayed. The previous display is no longer on the CRT. As illustrated at the end of the CRT display, the user then selects R | RETURN to return to his original display.

L or LOAD - During some utilization of MMS, the user may have had to stop during execution of one of the packages , as SGL. Because the user did not want to loose all the time and efforts invested so far, he has done a U or UNLOAD. Registers, tables, etc., are saved plus necessary parts of memory. Thus, during L | LOAD, all of these saved parts are reloaded from this disk area. The involved MMS software package will have to ask the user for the name of this disk file in order to accomplish this restoration.

N or NO - This response is acceptable only where indicated.

P or PAUSE - This response is acceptable at any time by any of the user-interactive programs. The user can process the PAUSE as a DEBUG|PMS breakpoint if the MMS program is RTE. In all cases, the two acceptable responses are C|CONTINUE or U|UNLOAD.

Q or QUICK - During SGL or RTE, the user wishes to utilize the package without having to be in the interactive mode.

R or RETURN - When using the interactive mode, the user may make a wrong selection. The R|RETURN ignores the most recent choice and returns to the previous display and execution step. This option will have to be limited to apply only to the most recent display; i.e., a series of two or more R's cannot be handled by SGL or RTE.

S or SUMMARY - During the configuring of the TS within SGL or during the execution of RTE, the Summary Table can be displayed at any time. This shows the configuring of the PE's. A R|RETURN will return to previous display.

T or TERMINATE - This option causes control to go to the FCP Executive. Results are not saved, except for some updates to the library files which may have been noted to the user as being saved.

U or UNLOAD - After a P|PAUSE, the user may wish to stop and to be able to save enough data to continue execution of the program at this point. This option, if feasible, allows this condition. Note that this option should be included in MMS only if SGL or RTE happen to require a lot of time; otherwise, the cost required to implement this feature is not feasible.

Y or YES - This response is acceptable only where indicated.

As indicated at the end of the "Help" file, all other characters are do-nothing commands and will cause no action or any change of displays.

```
YOU NEED HELP.  FURTHER EXPLANATIONS ARE:
  C - CONTINUE; YOU CHANGED YOUR MOND AFTER SELECTING (P).
  D - DONE; YOU WISH TO TERMINATE MMS WITH TABLES UPDATED.
  H - HELP; YOU GET THIS "HELP" TABLE.
  L - LOAD; YOU PREVIOUSLY DID AN UNLOAD (U); NOW YOU WISH TO RE-LOAD AT THAT POINT
      AND CONTINUE SYSTEM GENERATION.
  N - NO.
  P - PAUSE; YOU WISH TO STOP AND THINK. THE RESPONSES:
              C - CONTINUE
              U - UNLOAD
  Q - QUICK; YOU WISH TO DO A QUICK VERSION OF THE OPTION AREA.
  R - RETURN; YOU CAN 'BACK UP' TO PREVIOUS LEVEL OR QUESTION
  S - SUMMARY; YOU CAN PRINT AN UPDATED SUMMARY TABLE.
  T - TERMINATE; YOU ARE GIVING UP; CONTROL GOES BACK TO FCP EXECUTIVE
      AND RECENT RESULTS  ARE NOT SAVED.
  U - UNLOAD;  YOU WISH TO STOP AT THIS POINT WITHIN MMSSGL WITH
      EVERYTHING SAVED.  A PAUSE (P) MUST HAVE BEEN DONE.  A LOAD
      (L) MUST BE DONE TO RECOVER.
  Y - YES.

  ALL OTHER CHARACTERS PRODUCE A DO-NOTHING COMMAND.

  ••••••••••••••••••••

  TYPE RETURN TO GO BACK TO PREVIOUS PRINTOUT.

  ••••••••••••••••••••
```

FIGURE 21.1 "Help" File

21.2 SGL (System Generation Loader)

Figure 21.0-2 illustrates SGL. The purpose to the user is to allow him to configure the emulators into PE's so that the user can execute TS software on the emulated modules.

To enable the beginning user to quickly utilize SGL without having to learn a new language or new commands, this program has an interactive mode. The user is lead through the configuration by choosing options of a menu. As the options are selected, they are stored within a table. The user does not have to wait for the fulfillment of the choices.

Section 21.2.1 will present the scenario of this beginning user. Section 21.2.2 will then present the scenario for a user who has more knowledge of SGL and who doesn't need to configure on an interactive basis. Section 21.2.3 illustrates the utilization of SGL on a batch basis; thus, he can configure and execute MMS during non-peak periods. Section 21.2.4 illustrates how to select an existing Target System configuration.

21.2.1 Interactive Mode

The interactive mode helps the user to select emulator packages from a library. The library allows the user to select the basic emulator packages, already configured computer systems, or one of the already configured Target Systems. Within this example, the user wishes to configure the Target System covered within Section 20.

Only fourteen basic displays are utilized to illustrate this example. As more details are identified relative to the user question area of all emulator packages (see Section 14 and 20), this scenario will have to be expanded.

Figures 21.2.1-1 to 21.2.1-25 will go through the user scenario to interactively configure the TS of Figure 20.2.

```
WHICH TARGET SYSTEM DO YOU WISH
TO LOAD ?
     1  DEC-20 [V1]- ALLEN
     2  DEC-20 [V0]- JONES
     3  NONE

**********
SELECT ONE OF THE ABOVE NUMBERS
OR  HELP, LOAD, PAUSE, QUICK, TERMINATE.
**********
3
```

INTERACTIVE MODE SCENARIO

FIGURE 21.2.1-1

```
WHICH COMPUTER DO YOU WISH TO LOAD?
     0  INTERCONNECTOR
     1  8080 [V15] -ALLEN
     2  DEC-20 [V19]- GEAR
     3  PDP- 11/45 [V3] -JONES
     4  OTHER

**********
SELECT ONE OF THE ABOVE NUMBERS
OR HELP, DONE, PAUSE, QUICK, RETURN, SUMMARY, TERMINATE
**********
3
```

INTERACTIVE MODE SCENARIO

FIGURE 21.2.1-2

YOU HAVE SELECTED PDP-11/45 [V3]-JONES

THE CONFIGURATION IS:

11/45- IEC

11/45 - TAPE ET [V2]- SMITH

11/45 - DISK ED [V6]- JONES

****

IF THE ABOVE IS NOT CORRECT, TYPE OTHER.

IF THE ABOVE IS CORRECT, DO YOU WISH TO SPECIFY THE PE NUMBER? TYPE YES OR NO.

OTHER ACCEPTABLE RESPONSES: HELP, PAUSE, RETURN, SUMMARY, TERMINATE.

****

OTHER

INTERACTIVE MODE SCENARIO

FIGURE 21.2.1-3

```
WHAT IS THE NAME FOR THE NEW COMPUTER
SYSTEM?

**********

TYPE COMPUTER NAME [VERSION]- YOUR NAME

**********

PDP-11/45 [V4] - MANN
```

INTERACTIVE MODE SCENARIO

FIGURE 21.2.1-4

```
YOU WISH TO CONFIGURE  PDP-11/45 [V4]- MANN

THE AVAILABLE COMPONENTS ARE:

      1   11/45- IEC
      2   11/45 - DISK ED [V6] -JONES
      3   11/45- TAPE ET [V2] - SMITH
      4   11/45-  BUS 6 [V8]- MANN
      5   11/45-  TAPE ET [V6] - MANN
      6   END


**********

TYPE NUMBER(S) FOR THE CONFIGURATION. 'END' OR ITS
      APPROPRIATE NUMBER WILL END THE SELECTION.

OTHER ACCEPTABLE RESPONSES: HELP, PAUSE, RETURN,
      SUMMARY, TERMINATE.

**********
      1 5 2 4 6
```

INTERACTIVE MODE SCENARIO

FIGURE 21.2.1-5

WHICH COMPUTER DO YOU WISH TO LOAD?

0 INTERCONNECTOR

1 8080 [V15] - ALLEN

2 DEC-20 [V20] - GEAR

3 PDP- 11/45 [V3] - JONES

4 PDP- 11/45 [V4] - MANN

5 OTHER

****

SELECT ONE OF THE ABOVE NUMBERS

OR HELP, DONE, PAUSE, QUICK, RETURN, SUMMARY, TERMINATE.

****

4

INTERACTIVE MODE SCENARIO

FIGURE 21.2.1-6

YOU HAVE SELECTED PDP-11/45 [V4]- MANN

THE CONFIGURATION IS:

11/45- IEC

11/45 - TAPE ET [V6]- MANN

11/45 - DISK ED [V6]- JONES

11/45 - BUS G [V3] - MANN

****

IF THE ABOVE IS NOT CORRECT, TYPE OTHER.

IF THE ABOVE IS CORRECT, DO YOU WISH TO SPECIFY THE PE NUMBER? TYPE YES OR NO.

OTHER ACCEPTABLE RESPONSES: HELP, PAUSE, RETURN, SUMMARY, TERMINATE.

****

YES

INTERACTIVE MODE SCENARIO

FIGURE 21.2.1-7

YOU HAVE CHOSEN TO SPECIFY THE PE NUMBER FOR
PDP -11/45 [V4] - MANN
THE PRESENT ASSIGNMENTS ARE:

NONE

AVAILABLE PE'S ARE:

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

****

WHAT NUMBER DO YOU WANT?

OTHER RESPONSES: HELP, PAUSE, RETURN, SUMMARY,
TERMINATE.

****
8

INTERACTIVE MODE SCENARIO

FIGURE 21.2.1-8

COMPUTER PDP-11/45 [V4]-MANN
IS BEING DOWNLOADED TO PE8.
DO YOU WISH TO RUN DIAGNOSTICS ON PE8
WHICH CONTAINS PDP-11/45 [V4]-MANN ?

****
TYPE YES OR NO.
OTHER RESPONSES: HELP, PAUSE, RETURN, SUMMARY, TERMINATE.
****

INTERACTIVE MODE SCENARIO

FIGURE 21.2.1-9

NO

```
WHICH COMPUTER DO YOU WISH TO LOAD?
     0  INTERCONNECTOR
     1  8080 [V15] -ALLEN
     2  DEC-20 [V20]- GEAR
     3  PDP- 11/45 [V3] -JONES
     4  PDP- 11/45 [V4]- MANN
     5  OTHER

**********
SELECT ONE OF THE ABOVE NUMBERS
OR HELP, DONE, PAUSE, QUICK, RETURN, SUMMARY, TERMINATE.
**********
```


INTERACTIVE MODE SCENARIO

FIGURE 21.2.1-10

1

YOU HAVE SELECTED 8080 [V15] - ALLEN

THE CONFIGURATION IS:

8080 - IEC

8080 - BUS G [V2] - MANN

8080 - SMN01 [V1] - SMITH

****

IF THE ABOVE IS NOT CORRECT, TYPE OTHER.

IF THE ABOVE IS CORRECT, DO YOU WISH TO SPECIFY THE PE NUMBER? TYPE YES OR NO.

OTHER ACCEPTABLE RESPONSES: HELP, PAUSE, RETURN, SUMMARY, TERMINATE.

****

YES

INTERACTIVE MODE SCENARIO

FIGURE 21.2.1-11

YOU HAVE CHOSEN TO SPECIFY THE PE NUMBER FOR

8080 [V15]- ALLEN

THE PRESENT ASSIGNMENTS ARE:

8

AVAILABLE PE'S ARE:

0 1 2 3 4 5 6 7 9 10 11 12 13 14 15

****

WHAT NUMBER DO YOU WANT?

OTHER RESPONSES: HELP, PAUSE, RETURN, SUMMARY, TERMINATE.

****

INTERACTIVE MODE SCENARIO

FIGURE 21.2.1-12

COMPUTER 8080 [V15]-ALLEN

IS BEING DOWNLOADED TO PE 9.

DO YOU WISH TO RUN DIAGNOSTICS ON PE

WHICH CONTAINS 8080 [V15]- ALLEN?

****

TYPE YES OR NO.

OTHER RESPONSES: HELP, PAUSE, RETURN, SUMMARY, TERMINATE.

****

NO

INTERACTIVE MODE SCENARIO

FIGURE 21.2.1-13

```
WHICH COMPUTER DO YOU WISH TO LOAD?
     0  INTERCONNECTOR
     1  8080 [V15] -ALLEN
     2  DEC-20 [V20]- GEAR
     3  PDP- 11/45 [V3] -JONES
     4  PDP- 11/45 [V4]- MANN
     5  OTHER

**********
SELECT ONE OF THE ABOVE NUMBERS
OR HELP, DONE, PAUSE, QUICK, RETURN, SUMMARY, TERMINATE.
**********
1
```

INTERACTIVE MODE SCENARIO

FIGURE 21.2.1-14

YOU HAVE SELECTED 8080 [V15] - ALLEN

THE CONFIGURATION IS:

8080 - IEC

8080 - BUS G [V2] - MANN

8080 - SMN01 [V1] - SMITH

****

IF THE ABOVE IS NOT CORRECT, TYPE OTHER.

IF THE ABOVE IS CORRECT, DO YOU WISH TO SPECIFY THE PE NUMBER? TYPE YES OR NO.

OTHER ACCEPTABLE RESPONSES: HELP, PAUSE, RETURN, SUMMARY, TERMINATE.

****

YES

INTERACTIVE MODE SCENARIO

FIGURE 21.2.1-15

YOU HAVE CHOSEN TO SPECIFY THE PE NUMBER FOR

8080 [V15]- ALLEN

THE PRESENT ASSIGNMENTS ARE:

8 9

AVAILABLE PE'S ARE:

0 1 2 3 4 5 6 7 10 11 12 13 14 15

****

WHAT NUMBER DO YOU WANT?

OTHER RESPONSES: HELP, PAUSE, RETURN, SUMMARY, TERMINATE.

****

INTERACTIVE MODE SCENARIO

FIGURE 21.2.1-16

7

COMPUTER 8080 [V15]-ALLEN

IS BEING DOWNLOADED TO PE7.

DO YOU WISH TO RUN DIAGNOSTICS ON PE

WHICH CONTAINS 8080 [V15]- ALLEN?

****

TYPE YES OR NO.

OTHER RESPONSES: HELP, PAUSE, RETURN, SUMMARY, TERMINATE.

****

NO

INTERACTIVE MODE SCENARIO

FIGURE 21.2.1-17

WHICH COMPUTER DO YOU WISH TO LOAD?

0 INTERCONNECTOR

1 8080 [V15] -ALLEN

2 DEC-20 [V20]- GEAR

3 PDP- 11/45 [V3] -JONES

4 PDP- 11/45 [V4]- MANN

5 OTHER

****

SELECT ONE OF THE ABOVE NUMBERS

OR HELP, DONE, PAUSE, QUICK, RETURN, SUMMARY, TERMINATE.

****

0

INTERACTIVE MODE SCENARIO

FIGURE 21.2.1-18

YOU HAVE CHOSEN TO SPECIFY AN INTERCONNECTOR.

THE AVAILABLE TYPES ARE:

1 BUS A [V2] - JONES

2 BUS B [V9] - JONES

3 BUS G [V5] - MANN

4 NONE - END

****

SELECT ONE OF THE ABOVE NUMBERS

OR HELP, DONE, PAUSE, RETURN, SUMMARY, TERMINATE.

****

INTERACTIVE MODE SCENARIO

FIGURE 21.2.1-19

3

THE INTERCONNECTOR SELECTED IS BUS G[V5]-MANN.

THE PRESENT ASSIGNMENTS FOR COMPUTERS:

PE 8 - PDP-11/45 [V4] - MANN

PE 9 - 8080 [V15] - ALLEN

PE 7 - 8080 [V15] - ALLEN

****

SPECIFY BY NUMBER (AS 0,1,2) WHICH PE'S YOU WANT ON THIS INTERCONNECTOR.

****

8 9 7

INTERACTIVE MODE SCENARIO

FIGURE 21.2.1-20

YOU HAVE CHOSEN TO SPECIFY AN INTERCONNECTOR.

THE AVAILABLE TYPES ARE:

1 BUS A [V2] - JONES

2 BUS B [V9] - JONES

3 BUS G [V5] - MANN

4 NONE - END

****

SELECT ONE OF THE ABOVE NUMBERS

OR HELP, DONE, PAUSE, RETURN, SUMMARY, TERMINATE.

****

INTERACTIVE MODE SCENARIO

FIGURE 21.2.1-21

4

SUMMARY OF TS

PE 8 - PDP-11/45 [V4] - MANN
11/45 IEC
11/45 - TAPE ET [V6] - MANN
11/45 - DISK ED [V6] - JONES
11/45 - BUS G [V3] - MANN

PE 9 - 8080 [V15] - ALLEN
8080 IEC
8080 - BUS G [V2] - MANN
8080 - SMN01 [V1] - SMITH

PE7 - 8080 [V15] - ALLEN
8080 IEC
8080 - BUS G [V2] - MANN
8080 - SMN01 [V1] - SMITH

BUS G IS CONNECTED TO PE8, PE9, PE7

****

Do YOU WISH TO SAVE THIS CONFIGURATION? TYPE YES OR NO.

****

YES

INTERACTIVE MODE SCENARIO

FIGURE 21.2.1-22

WHAT IS THE NAME OF THIS NEW TS ?

MMSTST [V1] - MANN (12/25/79)

INTERACTIVE MODE SCENARIO

FIGURE 21.2.1-23

TARGET SYSTEM MMS TST [V1] - MANN (12/25/79)

IS BEING SAVED ON DISK.

DO YOU WISH TO HAVE THE SUMMARY SAVED?

****

IF YES, TYPE NAME OF OUTPUT FILE.

IF NO, TYPE NO.

****

MANN.OUT

INTERACTIVE MODE SCENARIO

FIGURE 21.2.1-24

```
YOU WANT THE SUMMARY SAVED IN FILE MANN.OUT
ARE YOU DONE?

**********
TYPE YES TO TERMINATE.
TYPE NO TO CONTINUE TS SPECIFICATION.
**********
```

INTERACTIVE MODE SCENARIO

FIGURE 21.2.1-25

```
Y
```

21.2.2 Quick Mode

This mode allows the user to quickly specify his options without having to be in the interactive mode. When the first display is on the CRT, the user is told that one option is Q[QUICK. Figures 21.2.2-1 to 21.2.2-2 illustrates this scenario.

```
WHICH TARGET SYSTEM DO YOU WISH
TO LOAD ?
     1  DEC-20 [V1]- ALLEN
     2  DEC-20 [V0]- JONES
     3  NONE

**********
SELECT ONE OF THE ABOVE NUMBERS
OR  HELP, LOAD, PAUSE, QUICK, TERMINATE.
**********
QUICK
```

QUICK MODE SCENARIO

FIGURE 21.2.2-1

FOR QUICK LOAD, SPECIFY UNITS:

```
TS = MMSTST [V1]- MANN (12/25/79)
PE8 = PDP-11/45 [V4] - MANN
      11/45 IEC
      11/45 - TAPE ET [V6] - MANN
      11/45 - DISK ED [V6] - JONES
      11/45 - BUS G [V3] - ALLEN
PE9   = 8080 [V15]- ALLEN
PE7   = 8080 [V15]- ALLEN
INTER = BUS G [V5] - MANN: 8, 9, 7
FILE  = MANN.OUT
DONE
```


QUICK MODE SCENARIO

FIGURE 21.2.2-2

21.2.3 Batch Mode

This mode allows the user to create a "command" file and to execute from this file. A master command file could reference this command file for SGL and another command file for RTE. This procedure is standard for the FCP computer. Figure 21.2.3 illustrates this batch mode.

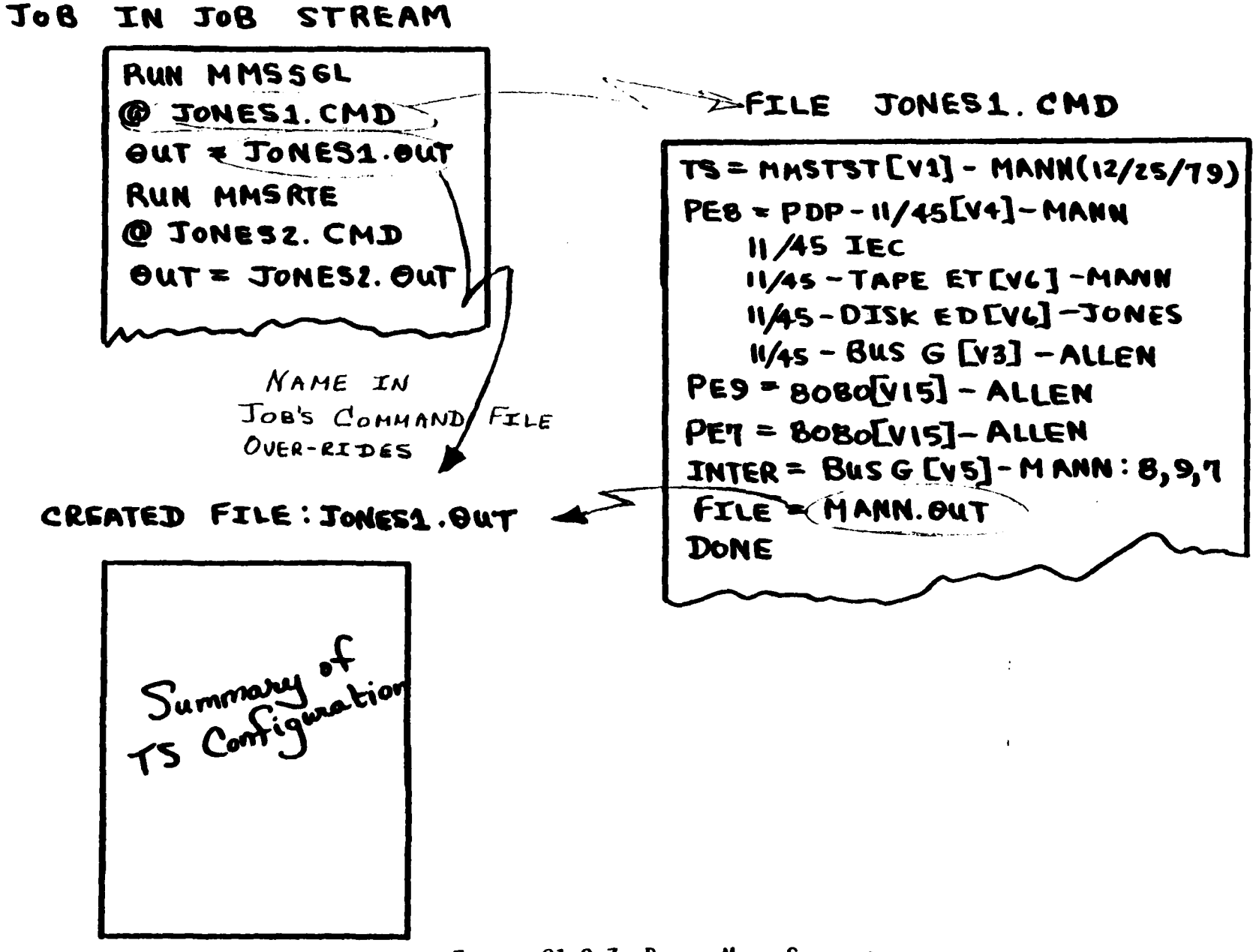


FIGURE 21.2.3 BATCH MODE SCENARIO

21.2.4 Loading of Existing TS

Once the user has configured his Target System and has saved this configuration in the TS Library, it is simple to reload the TS. Figures 21.2.4-1 to 21.2.4-2 illustrate this concept.

WHICH TARGET SYSTEM DO YOU WISH TO LOAD ?

1 DEC-20 [V1]- ALLEN

2 DEC-20 [V0]- JONES

3 MMSTST [V1]- MANN (12/25/79)

4 NONE

****

SELECT ONE OF THE ABOVE NUMBERS

OR HELP, LOAD, PAUSE, QUICK, TERMINATE.

****

3

TS QUICK MODE SCENARIO

FIGURE 21.2.4-1

```
THE CONFIGURATION FOR MMSTST [V1]- MANN (12/25/70) IS:
        PDP-11/45 [V4]- MANN
        8080 [V15]- ALLEN
        8080 [V16]- ALLEN
        BUS G [V5]- MANN


**********
DO YOU WISH TO SPECIFY PE NUMBERS?
        TYPE YES OR NO.
OTHER RESPONSES: HELP, PAUSE, RETURN, SUMMARY, TERMINATE.
**********
```


TS QUICK MODE SCENARIO

FIGURE 21.2.4-2

21.3 RTE (Run-Time Executive)

The modes of RTE are to be like those of SGL. The user will be able to execute his TS through the interactive use of RTE. With care, the user can process the RTE functions through the batch mode. Figures 21.3-1 through 21.3-8 are included to illustrate the variety of aids the interactive user obtains to help him to process RTE functions. Other displays will be required, but these will be developed as the design of RTE progresses.

```
RTE I/O CONFIGURATOR FOR COMPUTER SYSTEM
      PDP-11/45 [V4] - MANN

TO CONFIGURE I/O FOR  11/45 - DISK ED [V6] - JONES

      ENTER TYPE OF FILLER AND NAME OF FILLER

**********
FILLER TYPES ARE: DSK, MT, EXT, PGM
      FOR DISK, MAG TAPE, EXTERNAL DEVICE, PROGRAM
EXAMPLES ARE -  DSK: INP.DAT
                PGM: RUN.OBJ
OTHER OPTION ARE: HELP, DONE, PAUSE, QUICK, RETURN,
                  SUMMARY, TERMINATE
**********
```

RTE INTERACTIVE DISPLAYS

FIGURE 21.3-1

```
SELECT COMPUTER SYSTEM TO DEBUG.
CHOICES ARE:
        1- PDP-11/45 [V4] - MANN ON PE8
        2- 8080 [V15] - ALLEN ON PE9
        3- 8080 [V15] - ALLEN ON PE7
        4 - NONE
ALREADY CONFIGURED FOR DEBUG:
        3- 8080 [V15] - ALLEN ON PE7

**********
SELECT ONE OF THE ABOVE NUMBERS.
OTHER OPTIONS ARE: HELP, DONE, PAUSE, RETURN, SUMMARY,
                   TERMINATE, UNLOAD.

**********
```

RTE INTERACTIVE DISPLAYS

FIGURE 21.3-2

```
YOU ARE IN DEBUG FOR CS PDP-11/45 [V4]-MANN ON PEB.
ENTER NAME OF FILE WHICH CONTAINS SYMBOL TABLE.
        1- SYMBOL.DAT
        2- NO FILE AVAILABLE

**********
ENTER <1-> WITH NAME OF FILE OR <2> TO GO
      TO NEXT DISPLAY.
OTHER OPTIONS ARE: HELP, DONE, PAUSE, RETURN, SUMMARY,
                   TERMINATE, UNLOAD.
**********
```

RTE INTERACTIVE DISPLAYS

FIGURE 21.3-3

```
YOU HAVE SELECTED COMPUTER SYSTEM
      PDP-11/45[V4] MANN ON PE8 FOR DEBUG.
ENTER LEVEL OF DEBUG:
      1- HOL (NOT YET IMPLEMENTED)
      2- MACRO
      3- MICRO

**********
SELECT ONE OF THE ABOVE NUMBERS.
OTHER OPTIONS ARE: HELP, DONE, PAUSE, RETURN, SUMMARY,
                   TERMINATE , UNLOAD.
**********
```

RTE INTERACTIVE DISPLAYS

FIGURE 21.3-4

```
YOU ARE IN DEBUG FOR CS PDP-11/45 [V4]- MANN ON PEB.

ENTER MODE OF DEBUG:

        1- RUN-TIME
        2 - OFF-LINE
        3- BOTH

**********
SELECT ONE OF THE ABOVE NUMBERS.
OTHER OPTIONS ARE: HELP, DONE, PAUSE, RETURN, SUMMARY,
                   TERMINATE, UNLOAD.

**********
```

RTE INTERACTIVE DISPLAYS

FIGURE 21.3-5

SUMMARY OF DEBUG OPTIONS FOR PDP-11/45[V4]-MANN ON PE8.

ENTER MASTER OPTION SELECTION	SUMMARY (TYPE/VALUE)
1- HELP	1- B/ 01473 - ADDRESS
2- ABORT	2- B/ COST - SYMBOL
3 - BREAKPOINT	3- E/ OUTMP - SYMBOL
4 - COMPLETE - FINISHED WITH DEBUG	4- T/ Ø2577 - START TRACE
5- DELETE	5- T/ Ø261Ø - END TRACE
6- ERROR STOP	6-
7- INSPECT	7-
8- QUICK	8-
9- SUMMARY	9-
10 - TRACE	10-
	11-
	12-
	13-
	14-
	15-

****

SELECT ONE OF THE ABOVE NUMBERS TO SPECIFY TYPE, FOLLOWED BY /, FOLLOWED BY DELIMITER.

THE NUMBER 4 WILL TERMINATE SELECTION OF DEBUG OPTIONS.

OTHER OPTIONS ARE: HELP, DONE, PAUSE, RETURN, SUMMARY, TERMINATE, UNLOAD.

****

RTE INTERACTIVE DISPLAYS

FIGURE 21.3-6

HELP FILE FOR DEBUG

A- ABORT - ABNORMAL COMPLETION OF OPTION SELECTION; ALL OPTIONS SELECTED FOR THIS COMPUTER SYSTEM ARE IGNORED.

B- BREAKPOINT - POINT AT WHICH TO STOP EXECUTION.

C- COMPLETE - OPTION SELECTION FOR THIS CS IS FINISHED.

D- DELETE - REMOVE A SELECTED OPTION.

E- ERROR STOP - CONFIGURE ERROR TRAP.

H- HELP - DISPLAY THIS HELP FILE.

I- INSPECT- OBSERVE THE CONTENTS OF SPECIFIED ADDRESS.

Q- QUICK - NON-INTERACTIVE, "QUICK MODE" SELECTION OF DEBUG OPTIONS.

S- SUMMARY - DISPLAY SUMMARY OF DEBUG OPTIONS ASSIGNED.

T- TRACE - CONFIGURE A TRACE. FIST USE OF TRACE OPTION WILL SPECIFY START ADDRESS. THE NEXT USE OF TRACE OPTION WILL SPECIFY END ADDRESS. IF SECOND USE IS NOT INCLUDED, THIS WILL CAUSE TRACE UNTIL END OF EXECUTION OF PROGRAM.

****

R OR RETURN WILL CAUSE RETURN TO PREVIOUS SELECTION.

****

RTE INTERACTIVE DISPLAYS FIGURE 21.3-7

HELP FILE FOR DEBUG BREAKPOINT

A - ADDRESS - SET BREAKPOINT PRIOR TO ADDRESS EXECUTION.

C - COMPLETE - OPTION SELECTION FOR DEBUG BREAKPOINT IS FINISHED.

D - DELETE - REMOVE A SELECTED OPTION.

H - HELP - DISPLAY THIS HELP FILE.

I - INSTRUCTION - SET BREAKPOINT PRIOR TO EXECUTION OF SPECIFIED OPCODE.

T - TIME - SET BREAKPOINT AT SPECIFIED TIME (PSEUDO).

****

R OR RETURN WILL CAUSE RETURN TO PREVIOUS SELECTION.

****

RTE INTERACTIVE DISPLAYS

FIGURE 21.3-8

21.4 PMD (Post Mortem Dump)

The user scenario for PMD will be designed so that the user can interactively specify what areas or registers that he wishes to have displayed. The display can be in octal, decimal, hexidecimal, or as an opcode mnemonic. Because this can be a time-consuming process to scan parts of MMS, the user can dump major parts or all of MMS and the FCP computer to disk. This necessitates much available disk space for this operation, but the user can later scan and/or print parts of this dump.

21.5 PMS (PMS Off-Line Analysis)

The user scenario for PMS should be that the user can interactively specify the routines and then watch the display of the results. An option, after viewing the results, could be some printout. The user should be able to execute PMS in the batch mode. The results could then be as a printout or as a CRT display saved in a disk file. The user has the ability to select any of these disk display files for his CRT.

21.6 Abbreviations Used in Section 21

CRT	Cathode Ray Tube (user terminal)
FCP	Facility Control Processor
FCPI	FCP Interface
H/W	Hardware
IEC	Instruction Emulation Code
IOP	Input/Output Processor
MMS	Multiple Microprocessor System
Op Sys	Operating System
PE	Physical Element
PEXE	PE Executive
PMD	Post Mortem Dump
PMS	Performance Monitor System
RTE	Run-time Executive
SGL	System Generation Loader
Subs.	Subroutines
S/W	Software
TS	Target System

22.0 FCP DESCRIPTION

The FCP (Facility Control Processor) is to be a mini-computer. All software of MMS, except those specified as firmware, is to be either executable on the FCP or to be downloaded from the FCP. Therefore, all software of MMS has an impact upon the requirements of FCP.

SGL and RTE are executed on the FCP computer, and thus influence the requirements for execution times, memory size, and disk storage. The MMS Diagnostic routines require fast download time. The options of Post Mortem Dump require fast transfer of commands and of large blocks of data. PMD also requires a large disk area for saving full PE dumps, "Big 7" dumps, and the FCP dump at one time. Thus, these are the critical requirements for the FCP.

Three DEC mini-computers were evaluated to ascertain which one would best fit the needs of the MMS project. These three were: PDP-11/60, PDP-11/70, and VAX-780.

Since all source programs plus the object versions are to be saved on the disk of the FCP, this was one area of study relative to the FCP. All three PDP computers can utilize dual RPØ6's, or a more recent release. Each RPØ6 has a pack capacity of 176 Megabytes. This feature allows multi-user development, plus the growth of the necessary libraries. For the RPØ6:

Peak Transfer rate: 806 Kbytes/sec

Average Seek Time: 28 ms

Average Rotational Latency: 8.3 ms

Number Disk Drives/Controller: 8

At least one magnetic tape unit should be part of the FCP. This allows for the necessary system backup, should anything happen to the FCP. Again, all three can utilize the same unit. The TU-45 is program selectable 1600 or 800bpi, 9-track data storage. The read/write speed is 75 ips. Therefore, the data transfer rate is 120 Kbytes per second.

One line printer is mandatory. Because of the full MMS requirements, the line printer should be 1200 lines per minute with the 64-character set. A study of user requirements may show the necessity for the LXY Printer - Plotter. If much PMS plotting is indicated, this would provide smoother analysis graphs.

For I/O to the FCP computer, at least two CRT - type terminals are necessary. This allows interaction with some of the software packages, plus displays of results and graphs. An LA36 DEC writer could easily be included as part of the I/O so that the user of MMS can retain a copy of his decisions/input to MMS. The output as graphics would then be directed to the CRT, with all other output directed to either type.

Because the VAX-780 is a newer product, and is higher up the line than the PDP-11/60 and the PDP-11/70, it will have a longer life-span from today, although the other two are also fairly new products. The VAX can process an interrupt less than 10 micro seconds. If the PDP-11/70 is operating under RSX-11M, the VAX still processes the interrupt in about 2/3 of the time than does the PDP-11/70.

Also, because of the critical nature of predicted usage on MMS (relative to size of programs, development time, etc.), memory protection is a must. The VAX has this feature, as does PDP-11/70 under RSX-11M.

A sample performance test(designed and run by DEC) which consisted of two real-time tasks communicating on an event - driven basis shows the following:

PDP-11/70 with RSX-11M	1000 communications/sec (950 microsecs/trip)
VAX-11/780 with VAX/MMS	over 2200 communications/sec (450 micro secs/trip

The VAX-780 does have error-correcting memory. Thus if memory starts to become marginal in reliability, the program will still receive a good copy of data.

The VAX-780 has 32 interrupt priority levels. Sixteen of these are hardware and sixteen software.

Automatic restart is another good feature for ARPANET users or the general user. Both VAX and RSX-11M on the PDP-11/70 have this feature.

To help maintain the FCP computer, it is necessary to run diagnostics, especially if the FCP is down. The VAX has the feature that the diagnostics can be remotely processed over phone lines. This will greatly minimize down time.

Since much of the MMS will be developed in HOL (Higher Order Language), these speed ratios are important. The HOL could be ADA (PASCAL) or FORTRAN. Since FORTRAN measurements have been established, these are quoted. FORTRAN, in native mode, under VMS is 1.2-2.0 times faster than PDP-11/70 FORTRAN.

The VAX FORTRAN-IV-PLUS is a full implementation of ANSI '66 extended FORTRAN. This includes IF-THEN-ELSE and character data types.

The PDP-11/70 lacks large virtual memory address space. This is not true on the VAX. The VAX also has priviledged instructions, which will allow the better tuning of the FCP to meet the needs of vendor of MMS.

Many more arguments (or facts) could be stated to show comparisons of the three systems. The PDP-11/70 is better than the PDP-11/60, and the VAX-780 is better than the PDP-11/70. If MMS enters the "build phase" in one year and takes about 3½ years before the product is usable at RADC, the PDP-11/60 and PDP-11/70 are approaching obsolescence, if not past it in light of MMS requirements.

An effort of MMS-level work requires present state-of-the-art computer products for the FCP. DEC will probably have another computer system on the market within the next year. Therefore, FCP should be established as being something better than the best VAX and, therefore, something more costly than the best of VAX. MMS will not then be obsolete when it is built, but should be state-of-the-art for 5-10 years.

23.0 GENERAL TOPICS

Several topics are referenced within several sections (10.0 - 22.0) of this software document. Some of these did not logically fit within a section's discussion. Some were discussed in parts, but the parts were not functionally "tied together". Therefore, this section was added to include several miscellaneous topics which are of interest to the software design, whether these topics are functionally accomplished by hardware or by software.

23.1 Synchronous Busing Structure

SBS is the format of messages carried on the MMS bus. Therefore, this relates to any box which sends or receives from the bus. Within the PE, the box is the BIU.

At present, the format is that of 78 parallel lines with a 200 nanosecond transfer rate. The transfers are register-to-register. There are 30 Control lines and 48 Information lines. The final decision for number of lines is dependent upon the number of segments of MMS, and also upon the number of devices (PE's) per segment.

The functions of SBS are:

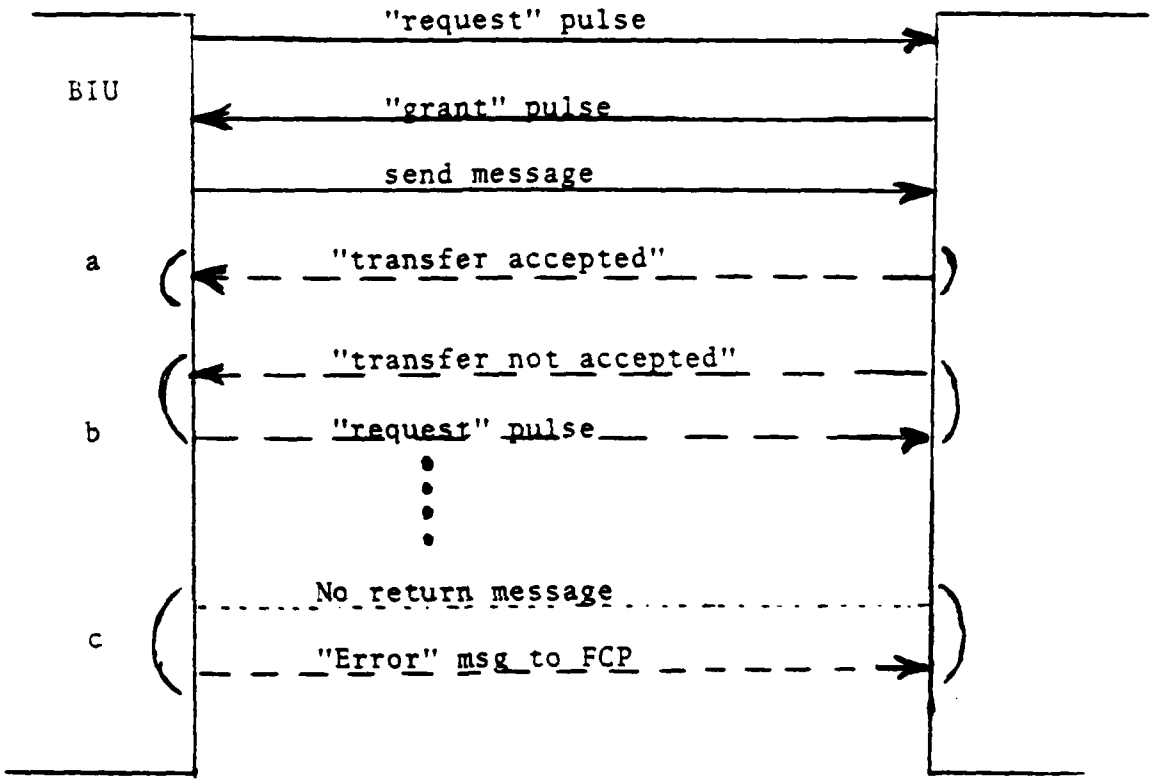
1) Control:

- 9 lines for Bus Timing and Handshaking
- 4 lines for descriptive control lines used by BIU
- 17 lines for Bus Request and Grant Lines

2) Information:

- 2 lines to describe if:
  - IPC (Inner Processor Communication)
  - BCC (Broadcast Communication)
  - Local Memory Address
  - PE Internal Memory Address
- 3 lines to identify Segment
- 5 lines to identify if either sub-address with PE or BIU address
- 22 lines to identify if either memory address or descriptive information
- 16 lines for data or for descriptive information

Message protocol (for sending a message) is:



Three conditions can result whenever a message is sent by the BIU within each PE.

23.2 Executive for IOP

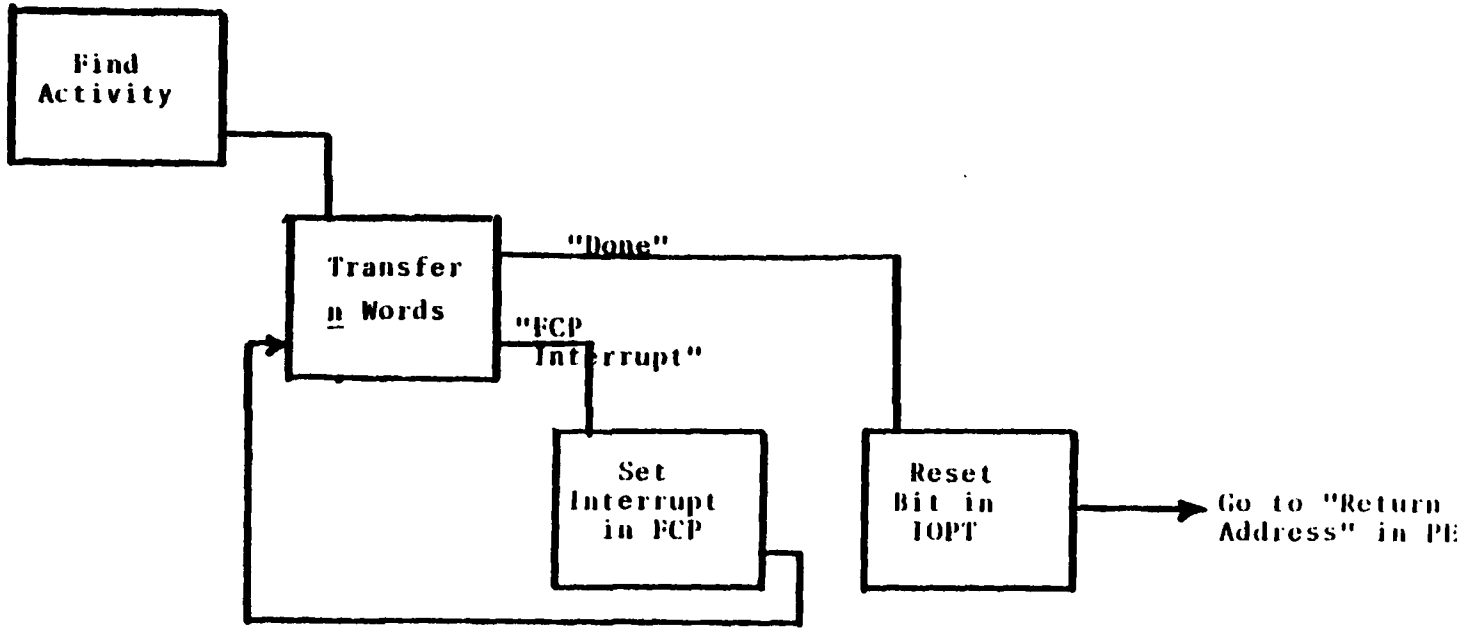
The Executive for the IOP was originally introduced in Section 10.2.3. This is to be "burned" into a PROM as firmware, and will be written in microcode.

The Executive will handle the transferring of data for the Target System. During SGL, the user specifies what emulators are required, and these are linked to the IOPT, which is an I/O linkage table in IOP. The user has also specified which method is to be used for this data transfer - either DMA or "time slicing". Both methods are part of PEXE.

During RTE, the user specifies the "other linkage" to the IOPT. Within FCP, this could be a program, a disk file, a mag tape, an external device, etc., to produce input data. Within FCP, the same types could exist to accept data from the TS. Another PE could be also utilized as the "other linkage", and the appropriate addresses would have to be generated and stored within the IOPT.

The Executive, when not busy transferring data, will be scaning the IOPT to find the activity with the highest priority. If the activity involves the transfer of a block of data, this amount of data is moved. If the data relative to the FCP involves the setting of an interrupt, this interrupt is set when necessary. Once the transfer is completed, a flag is set. The "ON" bit for this entry in the IOPT is reset.

Then, the Executive will wait for a "time out" from the Interval Timer. Then the "ON" flag in the IOPT for this activity is reset, the "return address" is sent to the PE, and TAC is started.



Executive

for IOP

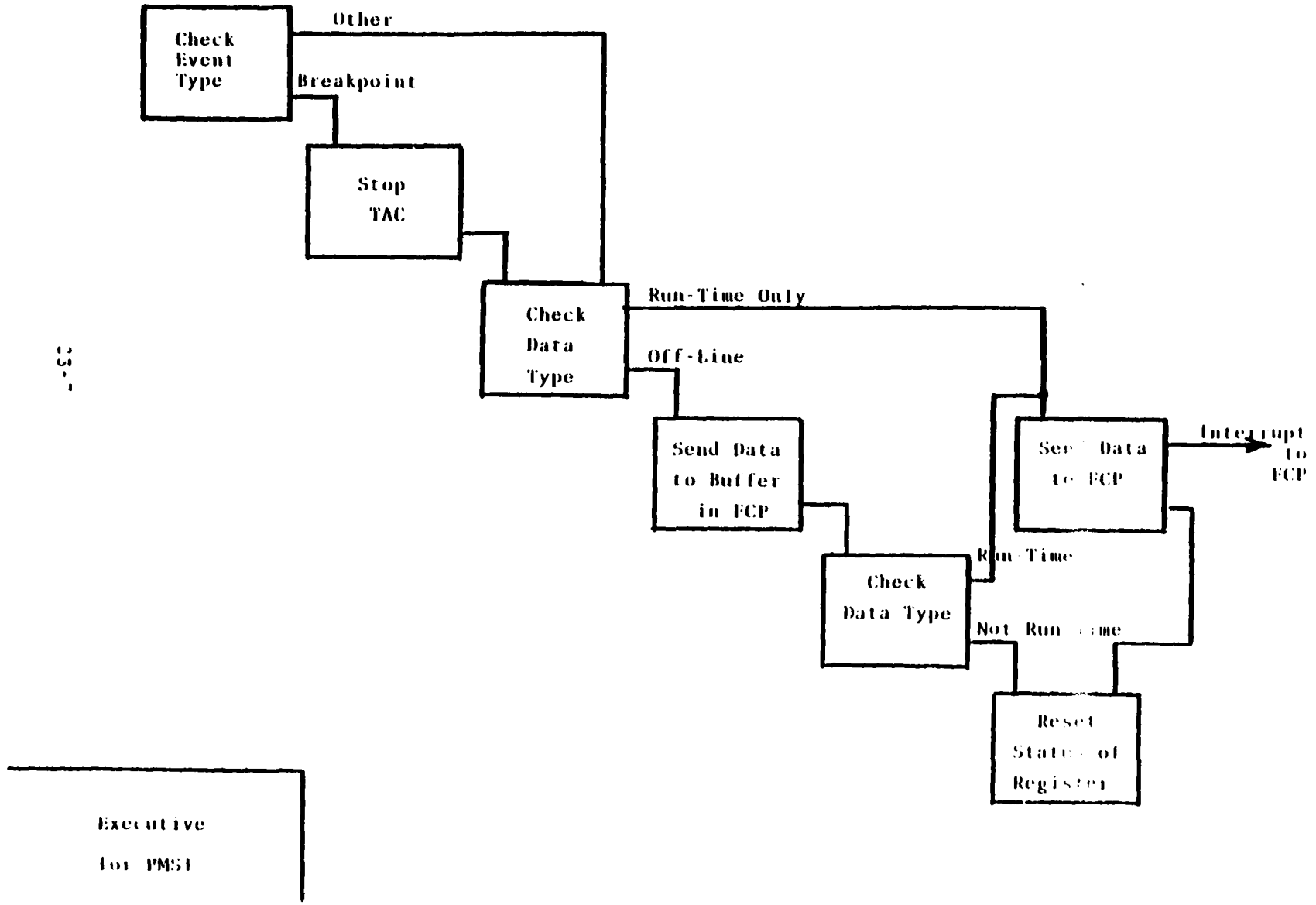
23.3 Executive for PMSI

The Executive for PMSI was originally introduced in Section 10.2.5. This routine is to be written in microcode and will be "burned" into a PROM.

The Executive will first check to see if the event, which caused the PMS/DEBUG data collection process, is a breakpoint. If so, TAC is stopped to allow the user to interact with the static condition of the TS. The data is sent to a buffer in FCP. An interrupt by PMSI to FCP will allow the PMS data routines to display the results on the user's terminal. The PMSI Status is changed to not busy. Once the user has finished, a continue will cause PEXE to resume control within the TS.

If the event is not a breakpoint, then the PMSI Executive will send the data to either one of the run-time/off-line buffers or to both the run-time and off-line buffers. Again, the off-line buffer system is dual-buffer with a utility routine in FCP to handle full buffers. Also, the run-time buffer is interrupt-driven; overflow cannot occur here because data is stored over previous data.

Once the data is sent to the FCP,the status of the PMSI Register is reset to allow new data to be accepted.


Check Event Type
Other
Breakpoint
Stop TAC
Check Data Type
Run-Time Only
Off-Line
Send Data to Buffer in FCP
Send Data to FCP
Interrupt to FCP
Check Data Type
Run Time
Not Run Time
Reset Status of Register
Executive for PMSI

23.4 Executive for SRC

The Executive for SRC was originally introduced in Section 10.2.6. This routine is to be written in microcode and will be "burned" into a PROM.

The SRC Executive will resolve arbitration for use of a shared resource. It has several ways to obtain the data for the requesting Computer System (CS).

Whenever a CS has requested to use a shared resource (SR), an entry has been placed in the Request Table. Therefore, the SRC Executive will scan this Table for a request of highest priority. If the process for transferring the data is multiplexing with the TS operations, then a data word and the value for arbitration delay is sent to the PE. Then the Executive waits for the next request from that PE before it sends another data set. Once the PE sends a release message, the Table is updated to reflect that this process is finished.

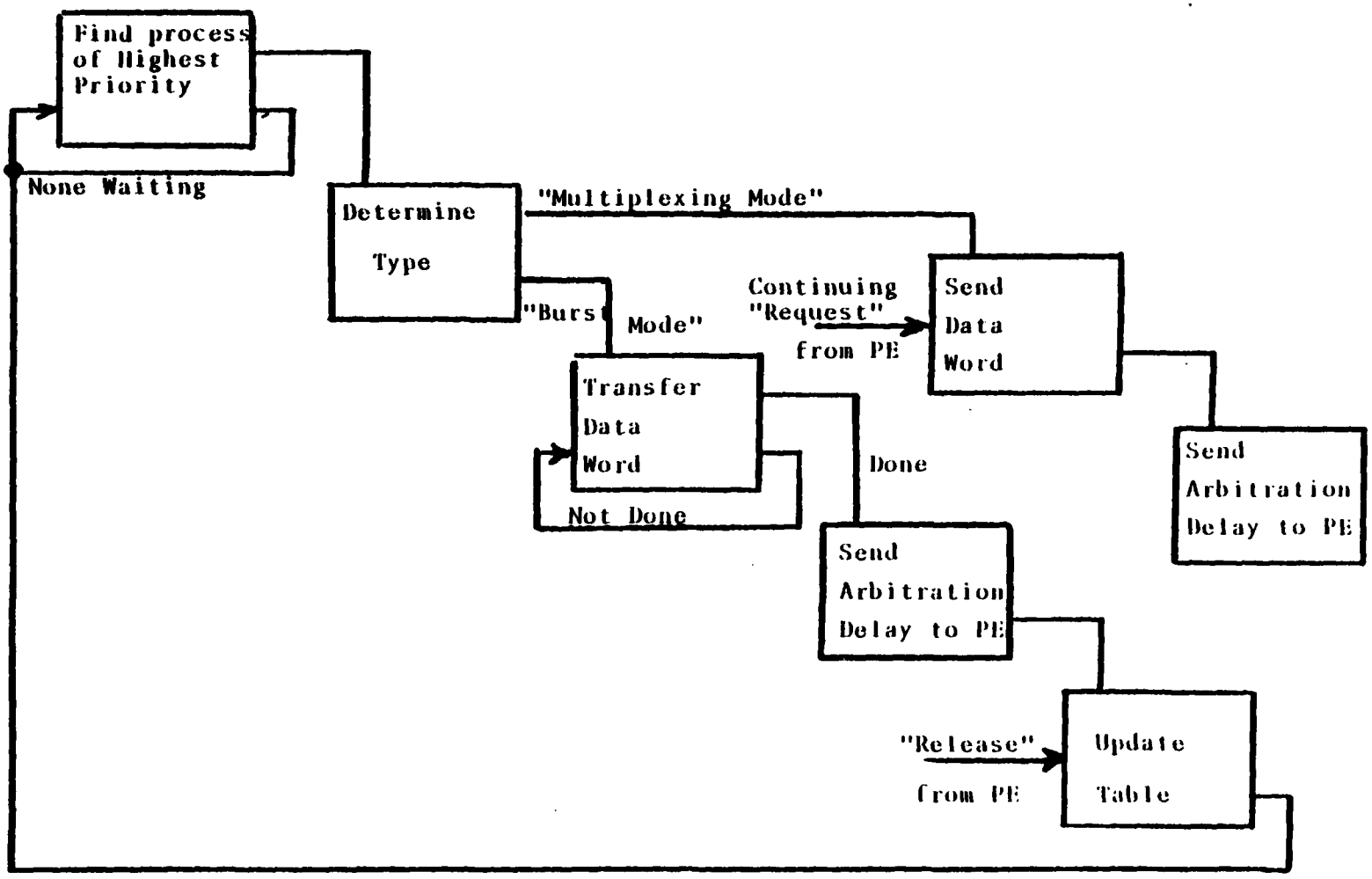
The difference between "burst" mode and "multiplexing" mode is that all data is transferred to the PE's memory, followed by the arbitration time. Again a release message has to be received by SRC.

The SRC Executive is to contain several possible ways to determine the arbitration for the emulated hardware. These methods are to be available to the user as possible schemes to implement.

During "burst" mode, the approach is for the hardware to get control and to retain that control until all accesses are satisfied. During emulation, the user should be able to select zero time, user-specified time, random-time, or the pseudo-time

needed to do the emulated process.

During "multiplexing", the accesses are time-sliced with the execution of other actions within the TS. The determination of time delay should be the same as that for the "burst" mode.



Executive for SRC

23.5 Abbreviations Used in Section 23

BCC	Broadcast Communication
BIU	Bus Interface Unit
CS	Computer System
DMA	Direct Memory Access
FCP	Facility Control Processor
IOP	Input/Output Processor
IOPT	Input/Output Port Table
IPC	Inner Processor Communication
MMS	Multiple Microprocessor System

MISSION
of
Rome Air Development Center

RADC plans and executes research, development, test and selected acquisition programs in support of Command, Control Communications and Intelligence (C^3I) activities. Technical and engineering support within areas of technical competence is provided to ESD Program Offices (POs) and other ESD elements. The principal technical mission areas are communications, electromagnetic guidance and control, surveillance of ground and aerospace objects, intelligence data collection and handling, information system technology, ionospheric propagation, solid state sciences, microwave physics and electronic reliability, maintainability and compatibility.

END

DATE
FILMED

8-80

DTIC